SEMI-ANNUAL STATUS REPORT

Error Control Techniques for Satellite and
Space Communications
NASA Grant Number NAG5-557

Principal Investigator:
Daniel J. Costello, Jr.

August 1994

Summary of Progress

In this report, we will focus on the results included in the Ph.D. dissertation of Dr. Yannick Levy, who was supported by the grant as a Research Assistant from August 1990 through May 1994. Dr. Levy completed his dissertation and received his Ph.D. degree in May 1994. A copy of the dissertation is included as Appendix A to this report. Two papers have already been published based on Dr. Levy's research [1,2]. In addition, one journal paper has been accepted for publication [3], another has been submitted for publication [4], and one more is in preparation for submission [5]. Finally, three conference presentations have resulted from this work [6-8]. The following sections contain brief summaries of this research.

1) Construction of Optimum Geometrically Uniform Trellis Codes

Since the publication of Ungerboeck's [9] pioneering paper on trellis coded modulation, many researchers have attempted to construct good codes and to analyze their performance. One of the major difficulties encountered is that, except in certain circumstances, the combination of a linear convolutional encoder with a mapping into a preselected signal constellation, such as 8-PSK or 16-QAM, results in a nonlinear trellis code. In other words, the set of distances between a reference code sequence and all other code sequences depends on the reference sequence chosen. This means that the distance properties of a given code cannot be evaluated by choosing the all-zero code sequence as the reference, unlike the situation with linear codes. Also, computer simulations of performance cannot assume that the all-zero code sequence was transmitted, as is routinely done for linear codes.

In a recent publication, Forney [10] outlined conditions, called geometric uniformity, under which a trellis code has exactly the same set of distances to other code sequences, independent of the reference sequence chosen. In other words, a geometrically uniform trellis code, although not necessarily linear, contains the essential property of a linear code needed to allow simplified code construction and performance analysis. Unfortunately, most previously constructed trellis codes are not geometrically uniform.

Several researchers have attempted to find good geometrically uniform trellis codes using a variety of approaches, including imposing only a group structure, rather than the usual finite field structure, on the code symbols and developing new methods of signal set partitioning. The key contribution of Dr. Levy's Ph.D. dissertation was the discovery of a totally new approach to constructing geometrically uniform trellis codes. In this approach, rather than following the standard procedure of preselecting a signal constellation, the signal parameters are treated as variables in the code construction algorithm, and a trellis structure, comprising the number of trellis states, the number of branches leaving each state, and the number of real valued code symbols on each branch, is preselected. This implies a fixed code rate and trellis complexity. Then a simulated annealing algorithm is used to assign code symbols to branches such that the free distance of the code is maximized, under the constraint that the code be geometrically uniform. The code symbols selected by the construction algorithm then define the signal constellation. In general, optimum free distance geometrically uniform

codes are obtained, and the signal constellations obtained are asymmetric. In many cases, improvements in free distance are obtained over codes previously believed to be optimum, under the assumption of a symmetric signal constellation.

Although this research had as its primary goal the construction of optimum geometrically uniform trellis codes, the approach used applies equally well to the construction of optimum binary convolutional codes. For, example, it is well known that the best rate 1/2, 4 state binary convolutional code has minimum free Hamming distance 5. If this code is used with QPSK modulation, the minimum free squared Euclidean distance is 10. Using the approach outlined above, a rate 1/2, 4 state code can be found using a rectangular, rather than square, 4-point signal constellation which has minimum free squared Euclidean distance 10.67, yielding about 0.3dB coding gain compared to the previously known "optimum" code. We believe this new approach to code construction has the potential to deliver fractional dB gains in performance compared to existing NASA coding systems with no increase in system complexity.

2) A Statistical Approach to Constructing Convolutional Code Generators

Another contribution made in Dr. Levy's Ph.D. research was the discovery of a relationship between the correlation coefficients of a set of convolutional code generators and the weights of the code sequences. This relationship allows one to develop an algorithm, again using simulated annealing, for constructing good, although in general suboptimum, convolutional codes with very large constraint lengths. For example, good rate 1/2 codes out to constraint length $\nu = 50$ have been constructed using this method. These long codes have potential in a sequential decoding system for achieving virtually error-free communication.

3) Calculating the Exact Performance of a Convolutional Code

Appendix B of this report contains a paper [3] recently accepted for publication by the IEEE Transactions on Information Theory. In this paper, a new technique is developed for calculating the exact bit error rate (BER) of a convolutional code. A Markov chain method is used to model a Viterbi decoder, following an approach used successfully to calculate the exact distortion of the Viterbi algorithm when used as a source encoder. Although this technique is applicable only to very short codes, it may be useful for computing the exact BER of the inner code in a concatenated coding system, since low complexity inner codes are often chosen for concatenated systems. Using performance bounds or time consuming computer simulations to estimate the BER of the inner code, on the other hand, can often lead to inaccurate results in determining overall concatenated system performance.

References

[1] Y. Levy, D. J. Costello, Jr., and A. R. Calderbank, "A Markovian Method Common to Both Quantization and Decoding Using Convolutional Codes", pp. 161-166, in Coding and Quantization, R. Calderbank, G. D. Forney, Jr., N. Moayeri (Eds.), American Mathematical Society, 1993.

[2] Y. Levy and D. J. Costello, Jr., "An Algebraic Approach to Constructing Convolutional Codes from Quasi-Cyclic Codes", pp. 189-198 in Coding and Quatization, R. Calderbank, G. D. Forney, Jr., N. Moayeri (Eds.), American Mathematical Society, 1993.

[3] M. R. Best, Y. Levy, P. C. Fishburn, A. Rabinovich, A. R. Calderbank and D. J. Costello, Jr., "An Exact Calculation of the Performance of a Convolutional Code", *IEEE Trans. Inform. Theory*, accepted for publication.

[4] Y. Levy and D. J. Costello, Jr., "Optimum Constellation Design for Geometrically Uniform Trellis Codes", *IEEE Trans. Inform. Theory*, submitted for publication.

[5] Y. Levy, and D. J. Costello, Jr., "A Construction of Convolutional Code Generators using Statistical Properties", *IEEE Trans. Inform. Theory*, to be submitted.

[6] Y. Levy and D. J. Costello, Jr., "A New Bound on the Free Distance of Rate 1/n Convolutional Codes", *Proc. IEEE International Symposium on Information Theory*, p. 139, San Antonio, TX, January 1993.

[7] Y. Levy and D. J. Costello, Jr., "A Systematic Construction for Geometrically Uniform Trellis Codes", *Proc. Conference on Information Sciences and Systems*, Princeton, NJ, March 1994.

[8] Y. Levy and D. J. Costello, Jr., "On the Construction of Real Number Trellis Codes", *Proc. IEEE International Symposium on Information Theory*, p. 163, Trondheim, Norway, June 1994.

[9] G. Ungerboeck, "Channel Coding with Multilevel/Phase Signals", *IEEE Trans. Inform. Theory*, Vol. IT-28, pp. 55-67, Jan. 1982.

[10] G. D. Forney, "Geometrically Uniform Codes", *IEEE Trans. Inform. Theory*, Vol. IT-37, pp. 1241-1260, Sept. 1991.

Appendix A

Real Number Trellis Codes

REAL NUMBER TRELLIS CODES

A Dissertation

Submitted to the Graduate School

of the University of Notre Dame

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

by

Yannick Lévy

Ingénieur Supélec, France, and M.S.E.E.

Daniel J. Costello, Jr., Director

Department of Electrical Engineering

Notre Dame, Indiana

April, 1994

REAL NUMBER TRELLIS CODES

Abstract

by

Yannick LEVY

Coding was first developed on the binary field. A redundant number of bits was added to the information in order to provide protection against noise, when transmitting or storing data. Recently, coding and modulation were combined in order to transmit data without requiring a lower transmission rate or a larger channel bandwidth. Although the construction was first done by combining binary codes and signal constellations in an efficient way, this type of coding scheme may also be viewed as coding on the field of real numbers.

The focus of this dissertation is the construction of trellis codes, since efficient decoding techniques are independent from the field in which codes are constructed. First, a new technique to construct binary convolutional codes based on the statistical properties of convolutional code generators is presented. The technique yields codes with much larger constraint lengths than previously constructed codes. In addition, the technique provides insights to aid in the design of good trellis codes on both the binary and real fields.

Asymptotic and non-asymptotic bounds on binary and real number codes are presented in a unified way in order to show the improvement expected by using real number trellis codes instead of binary trellis codes. This improvement is particu-

larly important for high rate codes, as well as for trellis codes with relatively short constraint length.

The usual scheme combining binary codes with expanded signal constellations is presented, and recent work involving multi-dimensional constellations, non-binary set-partitioning techniques, and the use of lattices to design signal constellations is described. This leads to some questions concerning the optimality of this design technique. A new technique based on the direct optimization of low rate trellis codes on the real field is then presented and some new codes show improvements over usual binary codes.

The problem of adjusting a large number of parameters when extending this construction to more complex rates and larger constraint length is then solved by adding an extra constraint in the design of real number trellis codes. Geometric uniformity is presented as an extension of the linearity concept for binary convolutional codes. It provides a way of designing the code with the same number of parameters than a binary convolutional code and to study its performance. New codes of various rates and constraint length are constructed, and it is proven that non-symmetric signal constellation are more adapted to the topology of trellis structures than the usual modulation schemes.

This concept provides a geometric view to the problem of designing codes in the Euclidean space, isomorphic to the real field. This new construction technique allows us to clearly decompose the problem of code design for low and high rate codes, and show why improvement is observed when constructing trellis codes in the real field as opposed to the binary field. The addition of shaping for high rate codes and the study of a general algebraic approach to describe our geometric construction is presented and left for future research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

First and foremost I would like to thank my advisor, Professor Daniel J. Costello, Jr., for his invaluable advice and help during our four years of collaboration. Working with Professor Costello at the University of Notre Dame has been a very enjoyable and interesting experience, and I hope other students will contribute to the "French connection" with his help in the future.

I would like to thank Diane Mills and Lance Perez for our fruitful discussions and their help concerning the English wording of this dissertation. Thanks also to Mark A. Herro, Ken D. Sauer, and Robert L. Stevenson for serving on my dissertation defense committee and helping me with some of the concepts introduced at the beginning of this dissertation.

Special thanks to my family in France, my friends around the World, and all the friends I made on the beautiful campus of the University of Notre Dame during my four years of studies. In particular, I would like to thank Barbara Schmitz for sharing an office with me in a perfect atmosphere.

I wish to acknowledge the financial support provided by the Société Anonyme de Télécommunications (S.A.T.) during the first part of my research, as well as their administrative support for doing my French national service as a researcher in an American University. I also wish to acknowledge the financial support provided by research assistantships from the National Science Foundation (NSF Grant NCR89-03429) and

the National Aeronautics and Space Administration (NASA Grants NAG5-557 and NAG3-1549).

Finally, I would like to thank the University of Notre Dame for accepting me into the Graduate Program, and providing a wonderful work environment. Also, thanks to the French Government for their interest in this research and the organization of different meetings with other students and professors in the United States.

CHAPTER 1

INTRODUCTION

1.1 Digital communications

The most important concept of the theory of communications is the conversion of analog signals into digital form. This transformation of a continuous signal in time or space into a discrete signal is done by sampling the analog signal in time or space, which is similar to what the human eye or ear does when receiving and transmitting a signal to the brain. While it is obvious that our eyes and ears do not have infinite precision, we do not seem to perceive a sampled version of what we see or hear, but rather have a continuous idea of images and sounds. This concept is used in communications when transmitting analog signals in digital form. Nyquist [1] proved that if a signal has a limited bandwidth, it is indeed possible to transmit that signal in a digital form without deterioration, which means that the entire analog signal can be recomposed from its samples. In a communication system, samples are quantized into a set of real values. An index pointing to the value is sent as a digital number. This process converts an analog signal into series of digital numbers as shown in Figure 1.1.

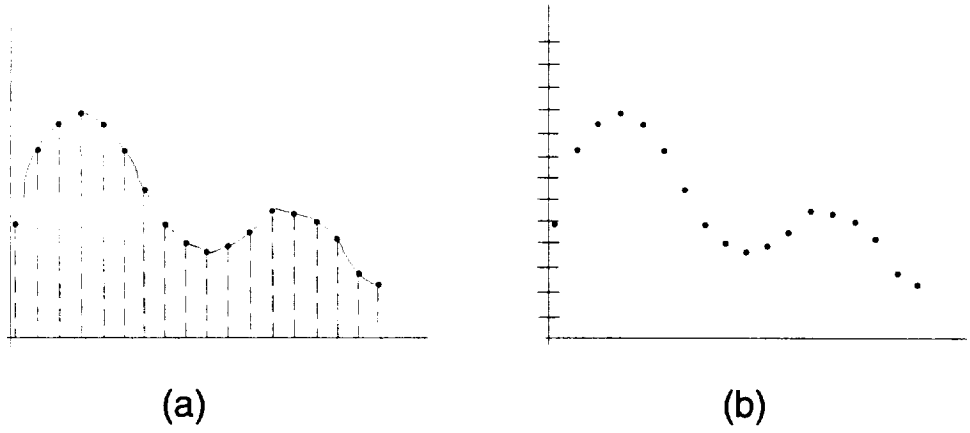

Figure 1.1: Digitalization of an analog signal: (a) Analog signal, (b) Sampled values.

There are several reasons for converting analog signals into equivalent digital signals. First, it is possible to multiplex different digital signals in the same transmission, that is, divide the entire transmission time into separate time slots used by different information sources. This is what communication companies are trying to do when sending data, voices, text and images through the same copper wires, satellite, optical or microwave links. Second, source coding techniques compress digital signals by removing redundancies, minimizing the transmission rate. Third, cryptography protects digital signals from undesirable listeners by changing the signification of each digits.

Finally, when compared to analog transmission, digital transmission as opposed to analog transmission efficiently protects the transmitted information from channel noise. If an analog signal is transmitted on a long distance, as on a satellite channel, or reamplified along the distance, as on telephone channels, the Signal to Noise ratio becomes too low for the receiver to understand the transmitted information. On the other hand, the digital form of the same signal can be transformed into a set of

patterns that the receiver can recognize. If the received signal is different from any of these patterns, the receiver can detect errors and may be able to correct them. This last concept is studied in coding theory, where one tries to design these patterns also called codewords, as well as efficient methods for encoding and decoding transmitted and received messages.

Coding theory not only provides techniques to correct messages corrupted by noise in transmission or storage, but also tries to minimize the energy needed to transmit as well as the bandwidth of the transmitted signals. The increasing volume of communications requires indeed to minimize the cost of transmissions mainly determined by the energy and bandwidth needed to transmit a signal. Error correction at a given energy was an early goal of coding theory, whereas increasing transmission speed over a channel with limited bandwidth started later with the combination of coding and modulation.

Engineers have always decomposed complex problems into distinct separable problems in order to solve the entire problem with simple solutions. Communications systems are typically modeled as shown in Figure 1.2. In 1949, Shannon [2] proved that some of the blocks in Figure 1.2 are separable asymptotically, that is, as the performance of each block becomes perfect, each block performance has been limited by its implementation. However, in the recent years, a new method of designing the blocks jointly has improved overall performance. The trend began in 1982 when Ungerboeck [3] combined channel coding and modulation, and continues now with the combination of source coding and channel coding for applications such as high definition television.

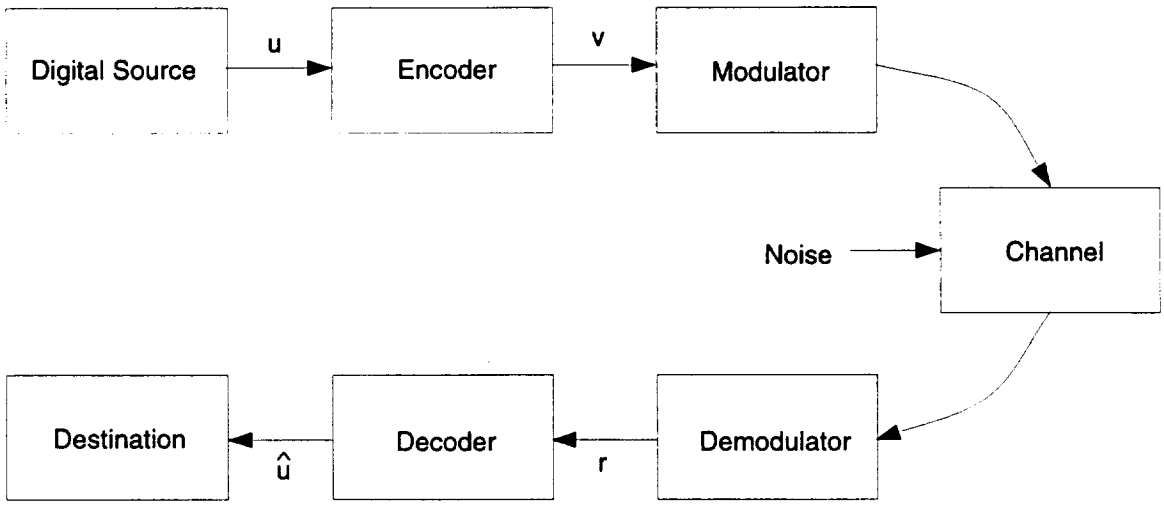

Figure 1.2: General communication system

In this dissertation, channel coding will often be seen as the entire block of combined coding and modulation and the optimization of this entire block will be sought. In order to describe coding theory better, the next sections will discuss signal modulation, present models used for the channel and channel noise, and summarize the state of the art in coding theory. This model will exhibit some important parameters that the coding theorist wants to optimize in the design of a communication system, and which will appear in the remainder of this dissertation.

1.2 Signal modulation

Although digital signals are often represented as series of digits, most wireless channels require the signal to be modulated before transmission. The modulation is a transformation of a digit into a function of time called waveform, and there is a one-to-one mapping between each possible digits and each waveform. These waveforms

are usually time-translated shaped pulses, and can be expressed as a sum of orthonormal functions $\phi_j(t)$ for t between 0 and T, T being the signaling interval, where two functions are orthonormal if and only if

$$\int_0^T \phi_j(t)\phi_l(t)dt = \delta_{jl}, \tag{1.1}$$

where $\delta_{jl} = 1$ if $j = l$ and $\delta_{jl} = 0$ if $j \neq l$. Orthonormal functions form the basis of an $N-$dimensional space. A signal expressed in this basis is an N-dimensional vector called a signal point and the set of signal points is called a constellation. Thus, each waveform can be represented as a signal point in an $N-$dimensional space.

Simple constellations are one and two dimensional constellations for which the orthonormal functions are cosines and sines with a given frequency. Constellations with larger dimensions have to be constructed using different frequencies or time intervals. For a given frequency F and a time interval T, two orthonormal functions can be derived:

$$\phi_1(t) = \sqrt{(\frac{2}{T})}\cos 2\pi F\frac{t}{T} \tag{1.2}$$

and

$$\phi_2(t) = \sqrt{(\frac{2}{T})}\sin 2\pi F\frac{t}{T}, \tag{1.3}$$

and any signal point s_i can be expressed as a two dimensional vector (s_{i_1}, s_{i_2}), and the transmitted waveform $s_i(t)$ is then expressed as

$$s_i(t) = s_{i_1}\phi_1(t) + s_{i_2}\phi_2(t). \tag{1.4}$$

More generally, for an $N-$dimensional constellation, any transmitted waveform can be expressed as an $N-$dimensional vector $(s_{i_1}, s_{i_2}, \ldots, s_{i_N})$, and the transmitted wave-

form $s_i(t)$ is then expressed as

$$s_i(t) = \sum_{j=1}^{N} s_{i,j}\,\phi_j(t). \tag{1.5}$$

One dimensional constellations are usually called PAM (Pulse Amplitude Modulation), and two dimensional constellations can be either QAM (Quadrature Amplitude Modulation), or PSK (Phase Shift Keying) for constant amplitude signal points. Some of these constellations are represented in Figure 1.3. The total number of signal points $\mathcal{N}$ corresponds to the total number of different sets of digits that can be sent during the time interval T. If digits are binary, that is taken from the field $F_2 = \{0, 1\}$, they are called bits, and $\log_2(\mathcal{N})$ bits can be transmitted at the same time. An $(N = 1, \mathcal{N} = 2)$ constellation can be either called 2-AM or BPSK (Binary Phase Shift Keying), and only one bit can be transmitted at a time.

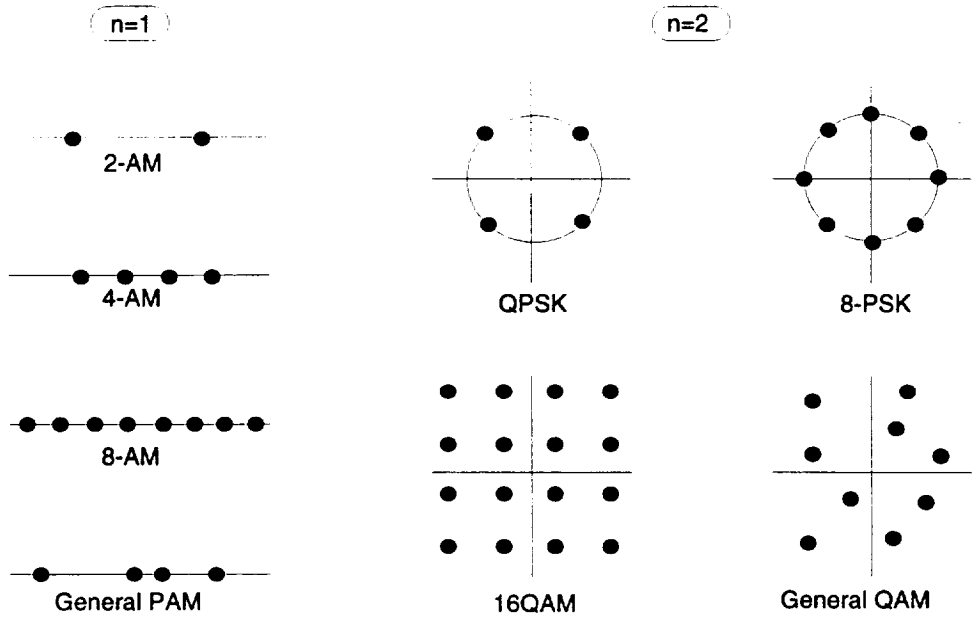

Figure 1.3: Different constellations in 1 and 2 dimensions.

Since the basis of each constellation is orthonormal, the energy required to send

a waveform corresponding to one of the orthonormal functions equals 1, where the energy E_f needed to send a function $f(t)$ is defined by

$$E_f = \int_{-\infty}^{\infty} f^2(t)dt. \tag{1.6}$$

Therefore, using the constellation, the energy E_f can also be computed as

$$E_f = |f|^2. \tag{1.7}$$

In order to compare different constellations, we need to normalize the energy needed per constellation use. This implies that the average energy to send all possible signal points of the constellation with equal probability is 1, that is

$$\frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} |s_i|^2 = 1. \tag{1.8}$$

In the next section, channels will be defined, and we will study how the constellation should be designed in order to achieve a high reliability when transmitting information.

1.3 Channel model

Channel is the general name given to any kind of transmitting or storing media. A channel can be copper wires, fiber optics, magnetic bands, disks, atmosphere or space. Different types of noise exist depending on the transmission media or the location. In order to design a reliable communication system, in which the receiver has the greatest probability of correctly interpreting a signal, one must determine a model for the channel corruptions. Two important channels with different characteristics exist in nature, and can be modeled as the AWGN (Additive White Gaussian

Noise) channel and the Fading channel. The Fading channel model is important in radiocommunications since it is usually a good model for moving transmitters and receivers in areas like cities, where waves rebound on buildings creating interferences. However, it is difficult to work with since its characteristics can vary a lot depending on the users and their situation. The AWGN channel model is simpler to handle and corresponds to satellite channels, telephone channels, high-frequency radio channels, magnetic tapes and disks. In the remainder of this dissertation, we will consider only the AWGN channel model, since it represents numerous real channels, and can be well protected against noise by using coding theory to increase transmission reliability.

If a signal $s(t)$ is transmitted over the AWGN channel, it is corrupted by an additive white Gaussian noise $n(t)$, and the received signal $r(t)$ can be expressed as

$$r(t) = s(t) + n(t), \tag{1.9}$$

where the white Gaussian noise has a Gaussian probability distribution, that is at a given time t_0,

$$Pr(n \leq n(t_0) \leq n + \epsilon) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{n^2}{2\sigma^2}), \tag{1.10}$$

where σ^2 corresponds to the variance of the noise equal to the one-sided power spectral density N_0 of the noise, and ϵ is a any given positive real number. Equations (1.9) and (1.10) can be written in the $N-$dimensional basis formed by the orthonormal functions defined previously. Thus,

$$r = s + n, \tag{1.11}$$

where r and s are the received and transmitted signals expressed in the $N-$dimensional

basis, and $n = (n_1, n_2, \ldots, n_N)$ is an $N-$dimensional noise vector statistically independent from s. More generally, in the remainder of this dissertation f will denote the signal point that expresses a given function of time $f(t)$. The *Signal to Noise Ratio* (SNR) is then defined as the average energy of a signal $|s|^2$ divided by N_0.

Upon reception of r, the receiver must decide which signal s was sent. The rule used by the receiver is a maximum a posteriori rule (MAP), which consists in deciding that s_i was sent if and only if $Pr(s/r)$ is maximum for $s = s_i$. Using Bayes's rule [4], this is equivalent to maximizing $Pr(r/s = s_i)$ assuming each signal to be sent equiprobably by the transmitter. This rule is called maximum likelihood estimation (ML). An ML receiver is optimum if all the messages are equally likely. Yet,

$$Pr(r/s = s_i) = p_n(r - s_i), \tag{1.12}$$

where $p_n(\alpha)$ is the $i.i.d$ $N-$dimensional Gaussian distribution of the noise with variance σ^2, that is

$$p_n(\alpha) = \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} e^{-\frac{|\alpha|^2}{2\sigma^2}}. \tag{1.13}$$

Thus, the ML receiver maximizes $e^{-\frac{|r-s_i|^2}{2\sigma^2}}$, which is equivalent to minimizing $|r - s_i|^2$. So the ML receiver decides that the closest signal point to the received signal point was sent. This corresponds to finding the signal point at the lowest Euclidean distance from the received point, where the Euclidean distance between two $N-$dimensional vectors $x = (x_1, \ldots, x_N)$ and $y = (y_1, \ldots, y_N)$ is defined by

$$d_E(x, y) = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}. \tag{1.14}$$

The squared Euclidean distance d_E^2 is often used instead.

As it was noted, the Euclidean distance plays an important role in optimal reception of signals transmitted over the AWGN channel. Since an ML receiver decides that the transmitted signal point is the closest to the received point in Euclidean distance, it appears interesting to maximize the distance between the signal points of a constellation such that there is as little confusion as possible for the receiver about which signal was transmitted, thus increasing the transmission reliability of the communications system. In the next section, coding theory will be introduced, and it will be shown that the Euclidean distance between signal points can be increased by grouping signal points together in order to increase the dimensionality of the space.

1.4 Coding Theory

Coding theory considers the design of blocks or sequences of digits, which once modulated on a signal constellation present a large Euclidean distance between each other. An ML decoder does not decode each received signal independently like it was assumed in the previous section but rather waits for the entire block or sequence of received signals. It was indeed shown by Shannon [2] and then studied in more details by Slepian [5] that a larger distance can be obtained by using longer blocks or sequences of digits. However, to avoid having to decrease the transmission rate, that is the amount of information per time unit, it is necessary to be able to send the same information during longer time intervals, which corresponds to increasing the total number of possible blocks or sequences. Presently, coding theory considers the design of block codes, for which digits are assembled in independent blocks, and trellis codes,

for which a unique sequence of dependent digits and arbitrary length is sent.

While block codes are simply the extension of the previous section to $n-$dimensional codewords with $n \geq N$ (n is generally a multiple of N), trellis codes were constructed later in order to increase the length of codewords without having to increase the design and decoding complexity. We will first describe block codes.

1.4.1 Block codes

A *block code* is a set of M n-dimensional codewords on a field F, where F can either be $F_2 = \{0, 1\}$, F_q the Galois Field of order q with q prime, or $\mathbb{R}$ the set of real numbers. Thus, a code represents a subset of F^n. The number of codewords M is usually a power of 2; so $M = 2^k$, where k is the number of bits of the information blocks associated to the codewords of length n. Therefore, there is a one-to-one mapping between the M possible information blocks of length k and the M codewords of length n. The rate of the code is then defined as

$$R = \frac{k}{n} = \frac{1}{n}\log_2(M). \tag{1.15}$$

For a binary block code, that is for $F_2 = \{0, 1\}$, it is possible to send each bit by using a BPSK constellation. If the distance that separates the two signal points in the BPSK constellation is 2, then the squared Euclidean distance d_E^2 between two codewords equals

$$d_E^2 = 4d_H, \tag{1.16}$$

where d_H is the *Hamming distance* between the two codewords in F_2, which is equal to the number of bits where the codewords differ, as shown in Figure 1.4. Formally,

the Hamming distance between two codewords $u = (u_1, \ldots, u_n)$ and $v = (v_1, \ldots, v_n)$ is given by

$$d_H(u, v) = \sum_{u_i \neq v_i} 1. \tag{1.17}$$

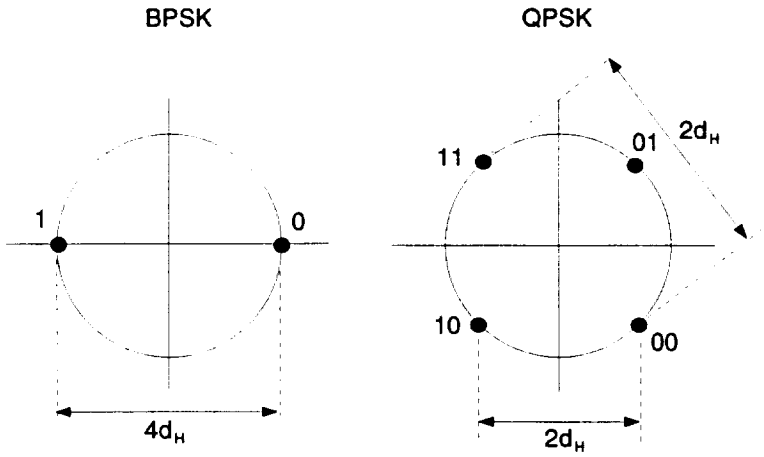

Figure 1.4: Relationship between Hamming distance and Euclidean distance for BPSK and QPSK constellations.

Similar to the use of the BPSK constellation, sending a codeword using a QPSK constellation requires to pair bits of the codeword, which allows us to send twice as many bits per constellation use. For the same energy per constellation use, the squared Euclidean distance is $d_E^2 = 2d_H$. Not only is the distance half the distance of a BPSK constellation, but the required energy is half the energy of a BPSK constellation because the QPSK constellation is used only half as many times as the BPSK constellation to send the same number of bits. Thus, increasing the number of points $\mathcal{N}$ in the constellation allows one to transmit the same codeword faster. However, increasing $\mathcal{N}$ without increasing the dimensionality of the constellation N cannot be done without losing the proportionality between the Hamming distance

and the Euclidean distance as in (1.16). In chapter 4, we will see techniques to map binary codewords on a constellation such that the Euclidean distance of the entire communication system increases with the Hamming distance of the binary code.

An important parameter associated with a block code is the *minimum distance* between any two codewords of this block code. Formally, the minimum Hamming distance of a code C is defined by

$$d_{min}(C) = \min\{d_H(u,v) : u,v \in C, u \neq v\}. \tag{1.18}$$

If a code is linear, that is, if the set of all codewords forms a vector space, then the minimum Hamming distance of a code is also equal to the *minimum weight* of any nonzero codewords, that is,

$$d_{min}(C) = \min\{w(v) : v \in C, v \neq 0\}, \tag{1.19}$$

where the weight $w(v)$ is the sum of all the nonzero components of v. It was noted that when the receiver decides which signal was transmitted, or when using coding, which codeword was transmitted, it decides of the closest codeword to the received vector. A sphere of radius r in n dimensions is defined as the set of $n-$dimensional points at a distance less than or equal to r from one $n-$dimensional point called center of the sphere. Thus, suppose that each codeword in a code C represents the center of a sphere of radius r, then for $r < [\frac{1}{2}(d-1)]$, all spheres are disjoint as shown in Figure 1.5. In that case, any received $n-$dimensional vector at a distance less than or equal to r from a codeword can be uniquely decoded. For the binary case, this means that at most r errors can be corrected. Thus, the greater the distance is, the larger the number of corrected errors can be.

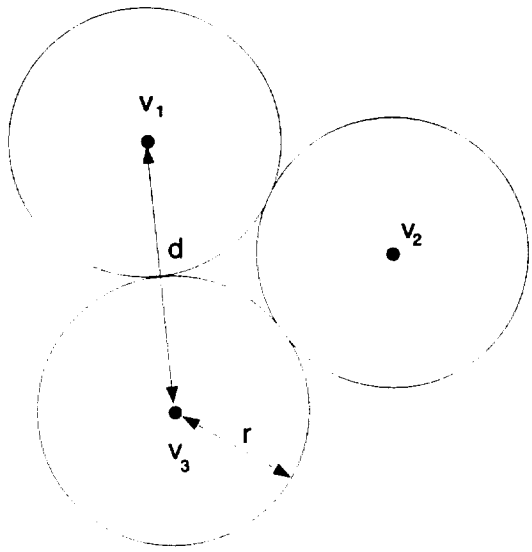

Figure 1.5: $n = 2$, $M = 3$ block codewords and their associated spheres.

1.4.2 Decoding of block codes

Although it was shown before that a ML receiver should decide that the transmitted codeword is the closest in Euclidean distance to the received word, this decoding procedure, called *soft decision* decoding, becomes tedious as the number of codewords M or the number of dimensions n gets large. Indeed, such a decoding procedure, called *table lookup*, would result in comparing the distances between the received word and all possible codewords.

Another technique called *hard decision* decoding consists of mapping each coordinate of the received word to a signal point of the signal constellation, yielding a received word in the same field F than the code field. In that case, an algebraic technique can be used to decode the received word, which is much simpler than a table lookup decoding procedure for numerous codes such as BCH, Reed-Solomon, or Reed-Muller codes [6, 7, 8].

An important parameter that characterizes a block code is the *probability of error*, P_e, resulting from the use of the code [9, 10, 11]. The probability of error that is usually of interest in is the information bit error probability, defined as the expected number of information bit errors divided by the total number of information bits. The determination of the probability of error is often difficult, and either simulations or upper bounds calculations are performed. Also, the probability of a codeword error P_M, defined as the probability that a decoded codeword is different from the transmitted one (yielding in general many information bit errors), is simpler to evaluate or bound in some cases. For example, if a binary block code is used on a Binary Symmetric Channel (BSC) (hard decision decoding), that is an AWGN channel for which the probability of receiving a 0 (1) when transmitting a 1 (0) is p as shown in Figure 1.6, the probability of m errors in a block of n bits is

$$P(m,n) = \begin{pmatrix} n \\ m \end{pmatrix} p^m (1-p)^{n-m}. \tag{1.20}$$

Therefore, since r errors can be corrected,

$$P_M \leq \sum_{m=r+1}^{n} P(m,n). \tag{1.21}$$

Another bound called union bound based on the fact that P_M cannot be greater than $M-1$ times the probability of erroneously decoding the transmitted codeword as its nearest neighbor (closest codeword in Hamming distance), which is at a distance $d_{min}(C)$ from the transmitted codeword. Thus,

$$P_M \leq (M-1) \sum_{m=[d_{min}/2]+1}^{d_{min}} \begin{pmatrix} d_{min} \\ m \end{pmatrix} p^m (1-p)^{d_{min}-m}. \tag{1.22}$$

A tighter upper bound called Chernoff bound yields

$$P_M \leq (M-1)[4p(1-p)]^{d_{min}/2}. \tag{1.23}$$

Although exact calculations of the probability of error involve the knowledge of the number of codewords situated at a distance d from the transmitted codeword for all possible $d \geq d_{min}$, also called *distance spectrum*, the previous upper bounds show the relationship between the probability of error and the minimum distance of a code. The larger the minimum distance is, the smaller the probability of error is for small p. This concept is general when using a coding scheme, that is, a code with large minimum distance yields asymptotically a smaller error probability, *i.e.* for large Signal to Noise ratio, or equivalently for small p on a BSC. However, the entire distance spectrum is needed to evaluate the non-asymptotic error probability, and often the minimum distance is not the only indicator of a good code at low Signal to Noise ratio. In particular, if the number of nearest neighbor codewords, τ, is large, the performance of the code becomes worse, since the decoder can confuse the transmitted codewords with many other neighbors. The *coding gain* represents the amount of energy gained by using a code versus using the equivalent uncoded system for the same probability of error, and the *asymptotic coding gain* represents the coding gain at large Signal to Noise ratio.

Decoding techniques are limited by their complexity as M and n increase. As it was said previously, the performance of the codes increases as n increases, and in order to keep the same transmission rate, k increases proportionally, thus making $M = 2^k$ increase exponentially. In order to solve this complexity problem, a subclass

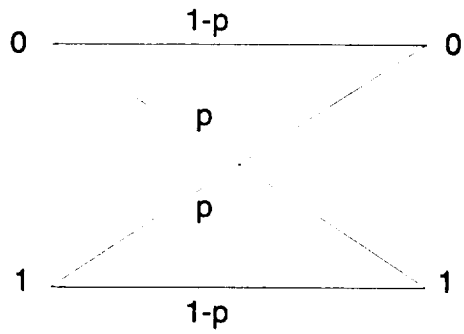

Figure 1.6: Transition probabilities for the Binary Symmetric Channel.

of infinite dimensional block codes, called trellis codes was discovered [12], for which block codewords become infinite code sequences. Due to the trellis structure which will be described in the next subsection, these code sequences can easily be decoded.

1.4.3 Trellis codes

A trellis code is a set of infinite dimensional vectors, called code sequences, related by a trellis structure, where a trellis structure is a finite state diagram expanded in time, as shown in Figure 1.7. The number of states σ in the trellis is usually a power of two, so $\sigma = 2^m$, where m is called the *constraint length* of the code. The information sequence consists of k-bits blocks which determine a path through the trellis. The code sequences consist of blocks of n branch labels, called encoded symbols, represented on each branch of the path on the trellis, chosen from a field F. Example 1.4.1 shows the example of an encoding procedure for an information sequence and the trellis code shown in Figure 1.7. The rate of the code is defined as

$$R = \frac{k}{n}. \tag{1.24}$$

Example 1.4.1 *In the trellis of Figure 1.7, consider the information sequence 1100. Starting from state 0, the first information bit 1 yields a transition from state 0 to 1, and produces the two information bits on this transition,* i.e., *11. Similarly, the following information bit produces 10, and the encoder jumps to state 3. Then, the information bit 0 produces 10 again, going back to state 2. Finally, the last information bit produces code bits 11, leading the encoder back to state 0. The complete code sequence associated to the information sequence 1100 is therefore 11101011.*

Binary linear trellis codes are called *convolutional codes* because they can be generated using a generator matrix, that is, the code sequences can be generated by performing a convolution between the information sequences and the generator matrix. Binary convolutional codes will be described into more details in Chapter 2.

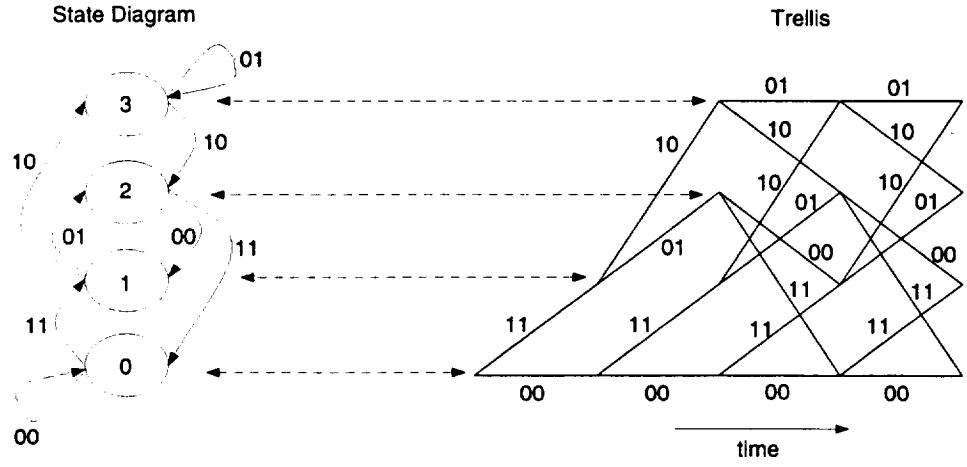

Figure 1.7: State diagram and trellis for a rate 1/2 constraint length 2 trellis code over F_2.

The *free distance* of a trellis code is defined as the minimum distance between any possible code sequences of the code. For a trellis code C,

$$d_{free}(C) \triangleq \min\{d(u,v) : u,v \in C, u \neq v\}, \tag{1.25}$$

where $d(u,v)$ can be the Hamming distance between two binary codes sequences u and v, or the Euclidean distance between u and v. Since a trellis has a finite number of states, it presents remerging paths, which can simplify the computation of the free distance, specifically,

$$d_{free}(C) = \min_{\substack{\lceil \frac{m}{k} \rceil + 1 \leq \tau \\ v_\tau \in C, v'_\tau \in e_{v_\tau}(C)}} d(v_\tau, v'_\tau), \tag{1.26}$$

where v_τ is a code sequence of length τ, and $e_{v_\tau}(C)$ is the set of all code sequences of length τ different from v_τ.

If a code is binary and linear, that is if the set of all code sequences forms a vector space, then the free distance of a code is also equal to the minimum weight of any nonzero code sequence, that is

$$d_{free}(C) = \min\{w(v) : v \in C, v \neq 0\}. \tag{1.27}$$

Thus, the free distance is also equal to the minimum weight of all code sequences of length τ, for τ greater than the shortest remerging path length, that is

$$d_{free}(C) = \min_{\substack{\lceil \frac{m}{k} \rceil + 1 \leq \tau \\ v_\tau \in C}} w(v_\tau). \tag{1.28}$$

Trellis codes were developed because their structure allows one to perform ML decoding without having to use a table lookup decoding procedure, or an algebraic technique which would prevent from using large sets of long codewords. This decoding

technique was first presented by Viterbi [13]. The next section explains the main idea of this technique which was shown to provide ML decoding.

1.4.4 Decoding of trellis codes

The Viterbi algorithm is an efficient algorithm for finding the path of length $\mathcal{L} = n(L + \lceil \frac{m}{k} \rceil)$ through the trellis which optimizes a *metric* between a received sequence and the sequence given by this path, where the metric can be either the Hamming distance for binary sequences, the Euclidean distance, or any expression of the distance that we are interested in. kL represents the length of the related information sequence to this path. The algorithm is as follows (see [9]):

- Step 1. At time unit $j = m$, compute the metrics for the single path entering each state. Store the metric for each state.
- Step 2. Increase j by 1, and compute the metric for all the paths entering a state by adding the branch metric to the previous connected state metric. Select the best path (path with the best metric), and store this path with its associated metric. Eliminate the other paths.
- Step 3. If $j < L + m$, repeat Step 2. Otherwise, stop and the path with the best metric gives the ML decoded sequence.

Example 1.4.2 *The trellis shown in Figure 1.8 represents the decoded path using Viterbi algorithm when the sequence 1010110111 is received, corresponding to a sequence of length* $\mathcal{L} = n(L + \lceil \frac{m}{k} \rceil) = 2(3 + 2) = 10$ *on a rate 1/2 binary convolutional*

code with constraint length $m = 2$. The metrics (Hamming distance) at each state are indicated on top of the state, and the paths with worse metrics than the best are crossed at each state for all time units greater than 2. The decoded path corresponding to the code sequence 0000110111 is shown in bold.

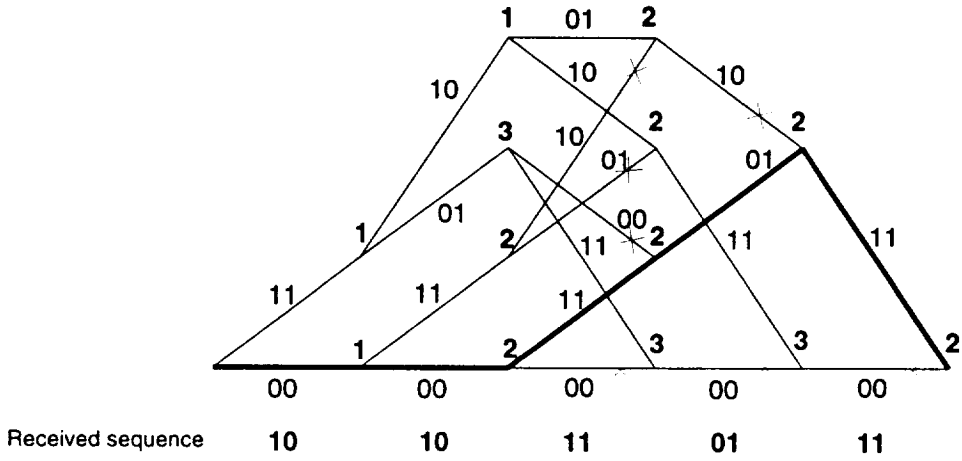

Figure 1.8: Viterbi decoding algorithm applied on a rate 1/2 constraint length 2 binary trellis code.

Since the metric can be the Euclidean distance, the Viterbi algorithm allows one to apply soft decision decoding on the received sequences. This advantage over block codes is important, since up to 3 dB can be gained on the asymptotic coding gain. Moreover, the decoding complexity only depends on the number of states in the trellis, not on the length of the sequences sent. One of the problems of the Viterbi algorithm as described previously is that one has to wait for the entire sequence to be received before the sequence can be decoded. In fact, it was shown that the decoding procedure can start once a sequence of about 5 constraint lengths has been received. Thus, as more bits of the sequence are received, the decoder can continuously decode,

thus reducing the total amount of memory needed for the decoder.

Again, we are interested in evaluating the probability of error when using a trellis code. The probability of first event error $P_f(E)$ for a binary convolutional code used on a BSC, that is, the probability that the decoded path diverges from the correct path for the first time can be upper bounded by

$$P_f(E) < \sum_{d=d_{free}}^{\infty} A_d P_d, \tag{1.29}$$

where A_d is the number of code sequences of weight d, and P_d is the probability of a first event error made on a weight d path. P_d can be computed [9] by noting that a first event error will be made if more than $d/2$ errors occur on an incorrect path, that is

$$P_d = \begin{cases} \sum_{e=(d+1)/2}^{d} \begin{pmatrix} d \\ e \end{pmatrix} p^e (1-p)^{d-e}, & d\,odd \\ \frac{1}{2}\begin{pmatrix} d \\ d/2 \end{pmatrix} p^{d/2}(1-p)^{d/2} + \sum_{e=(d/2)+1}^{d} \begin{pmatrix} d \\ e \end{pmatrix} p^e (1-p)^{d-e}, & d\,even. \end{cases} \tag{1.30}$$

Thus, by summing over all path of weight d for $d = d_{free}$ to infinity, the probability of first event error is upper bounded by

$$P_f(E) < \sum_{d=d_{free}}^{\infty} A_d [2\sqrt{p(1-p)}]^d. \tag{1.31}$$

For large Signal to Noise ratio, that is for small p, the bound is dominated by the free distance term, that is

$$P_f(E) \approx A_{d_{free}} 2^{d_{free}} p^{d_{free}/2}. \tag{1.32}$$

This bound can be transformed into a bound on the bit error probability $P_b(E)$,

$$P_b(E) \approx \frac{1}{k} B_{d_{free}} 2^{d_{free}} p^{d_{free}/2}, \tag{1.33}$$

where B_d is the total number of nonzero information bits on all weight d paths. Similar to the block codes case, the probability of error for large Signal to Noise ratio can be minimized by maximizing the free distance of the code, as well as minimizing the number of path with weight $d = d_{free}$, that is the number of nearest neighbors to the all zero sequence.

Note that some encoders, called *catastrophic* encoders, have an infinite number of paths with small distance. That is, for a catastrophic encoder, B_d or A_d are not finite, and the probability of error is not bounded. Although the minimum free distance of these codes can be as large as other "good" codes, the bad mapping between the information paths in the trellis and the code sequences yields this catastrophic effect. This catastrophic property can be simply eliminated for binary linear convolutional codes, as we will see in the next chapter.

We have seen in section 1.4.1 that if we use a BPSK or a QPSK constellation to send binary code sequences, the Euclidean distance between code sequences is proportional to the Hamming distance between them. However, using codes on these constellations may not be the best technique to achieve a large Euclidean distance at a given transmission rate. In fact, achieving a large distance may well require to use non binary code sequences, or equivalently other signal constellations. The next section will show with a simple example of a block code that non binary codewords can eventually reach a better Euclidean distance than binary codewords at the same

$\oplus$	0	1
0	0	1
1	1	0

$\times$	0	1
0	0	0
1	0	1

Table 1.1: Addition and multiplication in the binary field.

rate.

1.5 Construction fields for codes

Different fields can be used to construct codewords, the simplest one being the binary field. The binary field provides a multiplication and an addition described in table 1.1. The addition and multiplication are commutative, and the multiplication is distributive over the addition. This algebraic structure is very interesting for relating codewords and information words. The codewords can be easily derived from the information words, and the set of codewords becomes linear, which makes the analysis of the code much simpler, since the minimum distance can be computed by computing the minimum weight of nonzero codewords. Moreover, the addition between two bits indicates the Hamming distance between them. Numerous block codes can be derived by using algebraic techniques based on the binary field, such as Bose-Chaudhuri-Hockengheim (BCH), or Reed-Muller codes [6, 8].

More complex fields are q-ary fields $F = F_q$ with $q \neq 2$. These fields still provide a finite field algebraic structure, but often codewords have to be converted back to

a binary equivalent vector in order to be transmitted. This is the case for Reed-Solomon codes [7] for which codewords are constructed on fields with $q = 2^t$ with t integer strictly greater than 1. However, it was shown recently that similarly to the transmission of binary codes using BPSK or QPSK, it is possible to transmit codes over F_q using q-PSK constellation and keeping a proportional relationship between the Hamming distance over F_q and the Euclidean distance between transmitted codewords [14].

The most general field is $F = \mathbb{R}$. This field can provide the best distance between codewords at a given rate. However, the algebraic structure of the real field has not yet been useful in designing good codes. A method called coded modulation consists of designing binary convolutional codes and mapping binary outputs to real numbers. The mapping is done in such a way that a large Hamming distance for the binary code yields a large Euclidean distance for the resulting real code. This technique called *mapping by set partitioning* was first introduced by Ungerboeck [3], and is used a lot in present applications such as telephone modems or satellite communications, where the bandwidth, and therefore the rate is limited. This technique, however, only provides a subclass of all real codes, and extensions of these techniques will be described in Chapter 4.

The two block codes shown in Figure 1.9 demonstrate the difference between using real number codes and binary codes; Figure 1.9 shows two block codes composed of 3 two-dimensional codewords. Thus, the rate of both codes is $1/n \log_2(M) = 1/2 \log_2(3) = .79$. The binary codewords are restricted to four positions in the space : {-1,-1},{-1,1},{1,-1} and {1,1}, while the real codewords can be anywhere, thus

providing a larger minimum distance for the same average energy per code use.

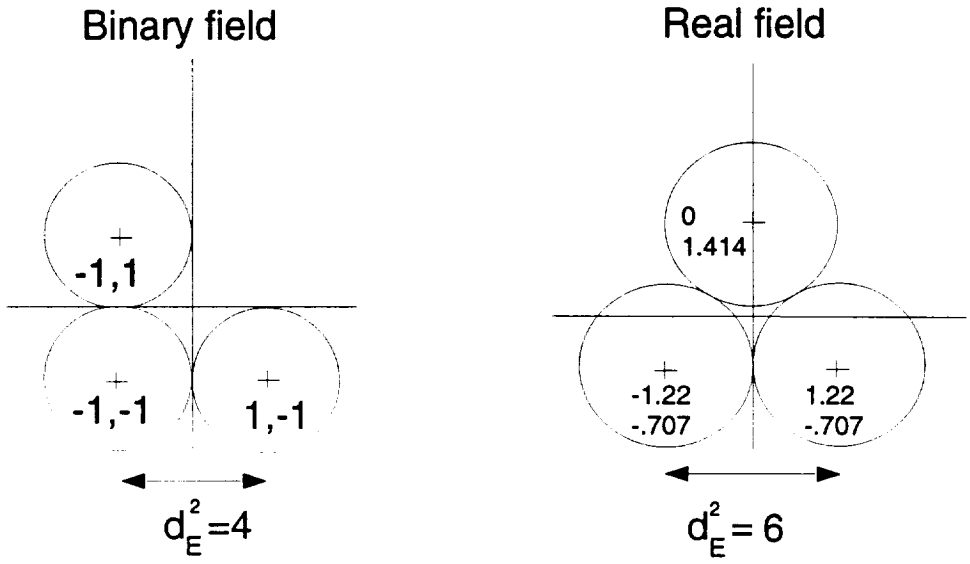

Figure 1.9: $M = 3, n = 2$ block codes in the binary and real fields.

The problem of designing real number codes is that there is no algebraic concept equivalent to the linearity on the binary field, which prevents us from simply computing the minimum distance by computing the minimum weight of nonzero codewords. Geometrically, this means that every codeword in the code does not have the same distance spectrum with its neighbors. The point of using coded modulation, that is a binary code mapped on a signal constellation, is that it is easier to control the linearity of the code by selecting a good mapping. New definitions for real number codes have been introduced over the recent years by Forney [15], and mapping between finite groups and signal sets have been proposed by Loeliger [16]. In this dissertation, these different concepts will be described in details, and the superiority of real codes over binary codes will be shown using bounds on the distance and code constructions. In the next section, an overview of the dissertation is presented.

1.6 Overview of the dissertation.

Historically, binary codes have been studied because of their easy implementation and algebraic structure. Some important code constructions and decoding techniques were found and implemented. Although the connection between a geometrical problem called *Sphere Packing*, which studies how many spheres of a certain radius can be packed in a given portion of the space, and the design of codes had been discovered in the very early ages of coding theory, only recent research has proposed using Sphere Packing to design codes and to bound the performance of codes. In particular, the Sphere Packing problem shows that binary codes are not the best codes that one can construct for a given transmission rate. On the other hand, the discovery of trellis codes has provided ways of performing soft decision decoding and increase the dimensionality of the coding space.

In the next chapter, we will develop some of the algebraic terminology used with binary convolutional codes, and the problem of constructing convolutional codes with a large free distance will be presented. A new approach for calculating the free distance of rate $1/n$ convolutional codes will be introduced, and new construction techniques, along with tables of some new binary codes, will be given. It will be shown that algebraic techniques first introduced for block codes can be extended to describe convolutional codes. However, other techniques are better to actually construct good codes (statistical in this chapter, and geometric later in the dissertation), as opposed to block codes for which algebraic techniques can lead to the construction of very good codes.

In chapter 3, the Sphere Packing problem and its relation to coding theory will be presented. Bounds on the distance of binary and real number block and trellis codes will be derived and plotted in order to give motivation to the reader for constructing real number codes as opposed to binary codes. This chapter will present a summary of all the best existing bounds on the distance of real and binary codes. These bounds will be compared to constructions later in the dissertation.

In chapter 4, we will introduce real number trellis codes based on Ungerboeck's decomposition of signal sets using partitioning, mapping, and underlying binary convolutional codes. Then, the basic improvements in Ungerboeck's techniques will be described, such as the use of lattices for constructing the constellation, multi-dimensional constellations as Cartesian products of lower-dimensional constellations. This will lead to the general definition of real number codes, and the construction of some low rate codes using simulated annealing on the $n2^{k+m}$ parameters of the code.

In chapter 5, the concept of geometric uniformity for real number trellis codes will be presented as an extension of the linearity concept for binary convolutional codes. This will lead to a new description of real number trellis codes using a decomposition of the trellis topology in order to construct a geometric structure leading to good geometrically uniform real number trellis codes. This decomposition leads to a construction algorithm for optimizing simultaneously the constellation and the code by using geometric considerations. Numerous codes are constructed for various rates, and improvements over usual trellis codes are noted.

Finally, the conclusion will summarize the different points of the thesis and review the new construction techniques and the codes presented in the dissertation. The

unification of the construction theory for binary and real number trellis codes using geometric considerations will lead to recommendations for future work such as the use of an algebraic techniques for executing geometric construction algorithms. Also, the concept of shaping will be shown to be applicable to the new codes with optimized distance in order to increase the overall gain.

CHAPTER 2

BINARY CONVOLUTIONAL CODES

Although binary convolutional codes are a subclass of trellis codes, they were discovered before the general class of trellis codes. Viterbi first used a trellis to represent the code sequences in order to implement the Viterbi algorithm [13]. In fact, binary convolutional codes are generated by a set of shift-registers that create a finite state machine, thus yielding a trellis structure.

2.1 Definition

A binary convolutional code is a trellis code with an encoder as represented in Figure 2.1 for which the connections can be described by the elements of a $k \times n$ generator matrix of binary polynomials. The generator matrix is given by

$$G(D) = \begin{bmatrix} g_0^{(0)}(D) & g_0^{(1)}(D) & \cdots & g_0^{(n-1)}(D) \\ g_1^{(0)}(D) & g_1^{(1)}(D) & \cdots & g_1^{(n-1)}(D) \\ \vdots & \vdots & \cdots & \vdots \\ g_{k-1}^{(0)}(D) & g_{k-1}^{(1)}(D) & \cdots & g_{k-1}^{(n-1)}(D) \end{bmatrix}, \tag{2.1}$$

where $g_i^{(j)}(D)$ for $i = 0 \ldots k-1$ and $j = 0 \ldots n-1$ is a binary polynomial. It is then possible to write the information sequences as a $k-$tuple of input binary polynomials $U(D) = (u^{(0)}(D), u^{(1)}(D), \ldots, u^{(k-1)}(D))$ and the encoded sequence is given by

$$V(D) = U(D)G(D), \tag{2.2}$$

where $V(D) = (v^{(0)}(D), v^{(1)}(D), \ldots, v^{(n-1)}(D))$ is a $n-$tuple of output binary polynomials.

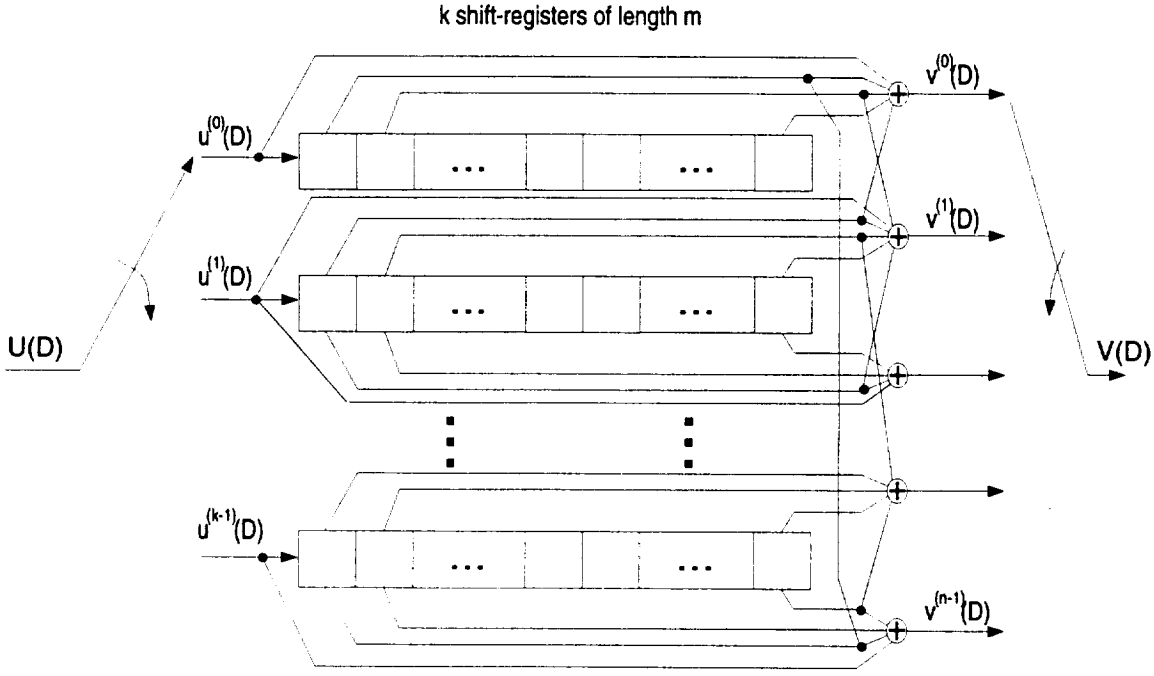

Figure 2.1: Binary rate k/n convolutional encoder.

An algebraic manipulation on $v(D) = v^{(0)}(D^n) + Dv^{(1)}(D^n) + \ldots + D^{n-1}v^{(n-1)}(D^n)$ yields

$$v(D) = \sum_{i=0}^{k-1} u^{(i)}(D^n) g_i(D), \tag{2.3}$$

where $g_i(D) = g_i^{(0)}(D^n) + Dg_i^{(1)}(D^n) + \ldots + D^{n-1}g_i^{(n)}(D^n)$ for $0 \leq i \leq k-1$ is called *composite generator polynomial* for the i^{th} input.

Equation (2.3) is particularly interesting for $k = 1$, *i.e.* for codes of rate $1/n$, since the code is completely defined by one composite generator $g(D)$ as follows

$$v(D) = u(D^n)g(D). \tag{2.4}$$

For further descriptions of convolutional codes, see [9, 11, 17].

Binary convolutional codes are interesting when trying to optimize the free distance, since the way they are generated makes them linear, which simplifies the computation of the free distance as seen in the introduction. In addition, the search for good codes is simplified to a search for good generators. However, the search for generators consists of searching for 2^m possible generators for each position in the generator matrix, which leads to $kn2^m$ possible generators. This limits the constraint length as well as k and n. The simplest rate in terms of number of generators is the rate $1/2$ convolutional code, for which exhaustive searches have so far lead to codes with constraint length up to $m = 18$.

Similarly to block codes, some algebraic techniques have been derived for constructing convolutional codes. Some techniques based on the construction using block cyclic codes [18] which present the same polynomial notation as convolutional codes have lead to codes that have a free distance far below the distance of optimum convolutional codes [19, 20] with same rate and constraint length. The problem associated with using cyclic codes to construct convolutional codes is that one cyclic code generator must be transformed into kn convolutional code generators. Since the weight of a code sequence is really the sum of the weights of the n code sequences in output, the n generators complement each other in order to produce the best possible

minimum weight. This interaction is difficult to realize when cyclic codes are used to construct convolutional codes. In order to avoid the problem of many generators, another algebraic construction was based on the use of quasi-cyclic codes [21, 22] which have exactly the same structure than convolutional codes, except that they are decomposed into finite blocks. However, these constructions did not lead to good convolutional codes either. These failed attempts to construct convolutional codes algebraically led researchers to call block codes algebraic codes, while convolutional codes are called more often probabilistic codes.

Based on this concept, we develop a new way of looking at the construction of generators for binary convolutional codes. This new idea tries to exploit the statistical properties of a good generator. In order to do so, we have to go back to the definition of the free distance, and try to incorporate the correlation coefficients of the generator in a new formula for computing the weight of a code sequence. In the next section, we will transform (2.4) and show that the weight (number of 1's) of a code sequence can be expressed as a function of the correlations of the generator and the information sequence.

2.2 On the weight of binary convolutional code sequences

In order to evaluate the free distance of a binary convolutional code, we have to compute the minimum weight of any code sequence. For rate $1/n$ codes, we have shown that a simple expression to find the code sequence from the information and

the composite generator is to use (2.4). Then the free distance simply becomes

$$d_{free} = \min_{u(D)\neq 0} w(u(D^n)g(D)). \tag{2.5}$$

Thus, the main problem in computing the free distance of binary convolutional codes is the difficulty of computing the weight of the product of two binary polynomials. The goal of this section is to introduce a new way of computing the weight of codewords. For this purpose, we derive a formula giving the weight of the product of two binary polynomials as a function of the correlations of the polynomials. First, we will evaluate the modulo-2 sum of a set of binary numbers, and then apply it to the coefficients of a product of polynomials.

2.2.1 On the weight of the sum of two binary numbers.

For the purpose of evaluating the weight of the sum of a set of binary numbers, we will evaluate the binary sum in the integer ring.

Lemma 2.2.1 *Let (x, y) be two elements from the binary field $F = \{0, 1\}$, let $\oplus$ ($\sum \oplus$ for large sums) denote addition in the binary field and $+$ ($\sum$ for large sums) denote addition in the integer ring I. Then,*

$$x \oplus y = x + y - 2xy. \tag{2.6}$$

Proof. *We can simply check (2.6) for every possible case:* $1 \oplus 1 = 1 + 1 - 2 = 0$, $1 \oplus 0 = 0 \oplus 1 = 0 + 1 - 0 = 1$, *and* $0 \oplus 0 = 0 + 0 - 0 = 0$.

We now generalize (2.6) to a set of n binary numbers:

Lemma 2.2.2 *Let $x_0, x_1, \ldots, x_{n-1}$ be n elements of F. Then,*

$$\sum_{i=0}^{i=n-1} \oplus x_i = \sum_{i=0}^{n-1} x_i - 2\left(\sum_{0=i<j}^{n-1} x_i x_j\right) + 4\left(\sum_{0=i<j<k}^{n-1} x_i x_j x_k\right) - \ldots \tag{2.7}$$

Proof. *The formula holds for n=2 (See (2.6)). Suppose (2.7) holds for n, i.e., for $x_0, x_1, \ldots, x_{n-1}$. Let x_n be an element of F. Then,*

$$\sum_{i=0}^{i=n} \oplus x_i = \left(\sum_{i=0}^{i=n-1} \oplus x_i\right) \oplus x_n. \tag{2.8}$$

By applying Lemma 2.2.1,

$$\sum_{i=0}^{i=n} \oplus x_i = \sum_{i=0}^{i=n-1} \oplus x_i + x_n - 2\left(\sum_{i=0}^{i=n-1} \oplus x_i\right) x_n. \tag{2.9}$$

Thus,

$$\sum_{i=0}^{i=n} \oplus x_i = \sum_{0=i}^{n} x_i - 2\left(\sum_{0=i<j}^{n} x_i x_j\right) + 4\left(\sum_{0=i<j<k}^{n} x_i x_j x_k\right) - \ldots \tag{2.10}$$

So (2.7) holds also for n+1, and therefore, by induction, (2.7) is satisfied for any n.

2.2.2 On the product of two binary polynomials

We now use (2.7) to derive a general formula on the weight of the product of two binary polynomials.

Lemma 2.2.3 *Let $a(D) = \sum_{i=0}^{i=\deg a} a_i D^i$, and $b(D) = \sum_{i=0}^{i=\deg b} b_i D^i$ where $a_i \in F$ and $b_i \in F$ for any i. Let $c(D) = a(D)b(D)$. Then,*

$$c(D) = \sum_{i=0}^{i=\deg a+\deg b} c_i D^i, \tag{2.11}$$

where [1]

$$c_i = \sum_{0=j<i}^{\deg a+\deg b} \oplus a_j b_{i-j}. \tag{2.12}$$

Proof. *By convolution of the two polynomials within F.*

For simplicity, we will use the notation $a(D) = \sum_{i=0}^{i=\infty} a_i D^i$ with $a_i = 0$ for $i > \deg a$. The weight of $c(D)$ is then given by the sum of its coefficients in I. So,

$$w(c(D)) = \sum_{i=0}^{i=\infty} c_i. \tag{2.13}$$

Thus,

$$w(c(D)) = \sum_{i=0}^{i=\infty} \left(\sum_{0=j<i}^{\infty} \oplus a_j b_{i-j} \right). \tag{2.14}$$

The first sum is in the integer ring, since we are adding all the ones in $c(D)$ to compute the weight. The second sum is in the binary field, since we are dealing with binary coefficients. We now transform the binary sum into an integer sum, by using (2.7). This yields the following lemma.

Lemma 2.2.4 *Let $a(D) = \sum_{i=0}^{i=\infty} a_i D^i$ and $b(D) = \sum_{i=0}^{i=\infty} b_i D^i$, where $a_i \in F$ and $b_i \in F$ for any $i > 0$. Let $c(D) = a(D)b(D)$. Then,*

$$w(c(D)) = \left(\sum_{i=0}^{i=\infty} a_i \right) \left(\sum_{i=0}^{i=\infty} b_i \right) - 2 \sum_{k=1}^{k=\infty} \left(\sum_{i=0}^{i=\infty} a_i a_{i+k} \right) \left(\sum_{i=0}^{i=\infty} b_i b_{i+k} \right) + 4 \ldots \tag{2.15}$$

Proof. *For a given $i \in I$, let $x_j = a_j b_{i-j}$ and use (2.7) to transform (2.14) as follows:*

$$\sum_{i=0}^{i=\infty} \left(\sum_{0=j<i}^{\infty} \oplus a_j b_{i-j} \right) = \sum_{i=0}^{i=\infty} \left(\sum_{0=j<i}^{\infty} a_j b_{i-j} - 2 \left(\sum_{0=j<l<i}^{\infty} a_j b_{i-j} a_l b_{i-l} \right) + 4 \ldots \right) \tag{2.16}$$

[1] deg means degree of the polynomial

This yields:

$$w(c(D)) = \sum_{0=j}^{\infty} a_j \sum_{0=j<i}^{\infty} b_{i-j} - 2 \sum_{0=j<l}^{\infty} a_j a_l \sum_{0=j<l<i}^{i=\infty} b_{i-j} b_{i-l} + 4 \ldots \tag{2.17}$$

By changing variables, (2.17) becomes

$$w(c(D)) = \left(\sum_{i=0}^{\infty} a_i \right) \left(\sum_{i=0}^{i=\infty} b_i \right) - 2 \sum_{k=0}^{\infty} \left(\sum_{i=0}^{\infty} a_i a_{i+k} \right) \left(\sum_{i=0}^{i=\infty} b_i b_{i+k} \right) + 4 \ldots \tag{2.18}$$

In order to express (2.15) differently, we need the following definitions of the correlation coefficients of polynomials:

Definition 2.2.1 *Let $a(D)$ be a polynomial with coefficients a_i. Then, we define the 0^{th} correlation coefficient of $a(D)$ by*

$$R_{0a} = \sum_{i=0}^{i=\infty} a_i, \tag{2.19}$$

the first correlation coefficients by

$$R_{1a}(j) = \sum_{i=0}^{i=\infty} a_i a_{i+j}, \tag{2.20}$$

where $j > 0$ and more generally the k^{th} correlation coefficients by

$$R_{ka}(j_1, j_2, \ldots, j_k) = \sum_{i=0}^{i=\infty} a_i a_{i+j_1} \ldots a_{i+j_1+\ldots+j_k}, \tag{2.21}$$

where $j_1, j_2, \ldots, j_k$ are k integers strictly greater than 0.

Using these definitions, we are able to write (2.15) differently, which leads to the following theorem:

Theorem 2.2.1 *Let $a(D) = \sum_{i=0}^{i=\infty} a_i D^i$ and $b(D) = \sum_{i=0}^{i=\infty} b_i D^i$, where $a_i \in F$ and $b_i \in F$ for any $i > 0$. Let $c(D) = a(D)b(D)$. Then,*

$$w(c(D)) = R_{0a} R_{0b} - 2 \sum_{j=0}^{j=\infty} R_{1a}(j) R_{1b}(j) + 4 \sum_{j,k=0}^{\infty} R_{2a}(j,k) R_{2b}(k,j) - \ldots \tag{2.22}$$

Proof. *Follows from Lemma 2.2.4 and definition 2.2.1.*

We now relate the weight of rate 1/n convolutional codewords to the formula obtained in Theorem 2.2.1, which will allow us to derive an expression for the free distance of these codes.

2.3 Weight of rate 1/n convolutional codewords

Let $g^{(1)}(D), g^{(2)}(D), \ldots, g^{(n)}(D)$ be the n generator polynomials of a rate 1/n convolutional code C. Then, let $g(D)$ be the composite generator as defined in (2.4). For any information sequence $u(D)$, the code sequence is generated by $v(D) = u(D^n)g(D)$. This leads to the following lemma:

Lemma 2.3.1 *Let $g(D)$ be the composite generator of a rate 1/n convolutional code. If $u(D)$ is the information sequence, then the weight of the code sequence $v(D)$ is given by*

$$w(v(D)) = R_{0u}R_{0g} - 2\sum_{j=0}^{j=\infty} R_{1u}(j)R_{1g}(nj) + 4\sum_{j,k=0}^{\infty} R_{2u}(j,k)R_{2g}(nk,nj) - \ldots \quad (2.23)$$

Proof. *Let $R_{ku}(j_1, j_2, \ldots, j_k)$ be the k^{th} correlation coefficient of $u(D)$, where $j_1, j_2, \ldots, j_k$ are k integers strictly greater than 0. Then the k^{th} correlation coefficient of $p(D) = u(D^n)$ is*

$$R_{kp}(j_1, j_2, \ldots, j_k) = \begin{cases} R_{ku}(\frac{j_1}{n}, \frac{j_2}{n}, \ldots, \frac{j_k}{n}) \text{ if } n \text{ divides } j_1, j_2, \ldots, j_k \\ 0 \text{ otherwise} \end{cases} \quad (2.24)$$

So, by using Theorem 2.2.1 and (2.4), we obtain:

$$w(v(D)) = R_{0_p}R_{0_g} - 2\sum_{j=0}^{j=\infty} R_{1_p}(j)R_{1_g}(j) + 4\sum_{j,k=0}^{\infty} R_{2_p}(j,k)R_{2_g}(k,j) - \ldots \quad (2.25)$$

Using (2.24), this yields

$$w(v(D)) = R_{0_u}R_{0_g} - 2\sum_{j=0}^{j=\infty} R_{1_u}(j)R_{1_g}(nj) + 4\sum_{j,k=0}^{\infty} R_{2_u}(j,k)R_{2_g}(nk,nj) - \ldots \quad (2.26)$$

This Lemma gives a useful way of computing the weight of a codeword. The following example helps us understanding the meaning of Lemma 2.3.1.

Example 2.3.1 *Let $u(D) = D^t$ with t an integer, $t > 0$. Then $R_{0_u} = 1$, and for any integer k, $k > 0$, $R_{k_a}(j_1, j_2, \ldots, j_k) = 0$. So,*

$$w(v(D)) = R_{0_g} = w(g(D)). \quad (2.27)$$

Let $u(D) = D^t + D^r$ with t and r integers, $r > t > 0$. Then

$$\begin{aligned} w(v(D)) &= 2R_{0_g} - 2R_{1_g}(n(r-t)) \\ &= 2w(g(D)) - 2R_{1_g}(n(r-t)). \end{aligned} \quad (2.28)$$

For example, with $g(D) = 1 + D + \ldots + D^s$, where s is an integer, $s > 0$,

$$w(v(D)) = 2(s+1) - 2(s-(r-t)+1) = 2(r-t) \quad (2.29)$$

if $(r-t) \leq s$.

This Lemma can also be used for the construction of codes with large free distance. First, we introduce two parameters related to the free distance of a trellis code. The *row distance* of order l of a convolutional code C is defined by the minimum weight

of code sequences up to length l. It is an upper bound to the free distance of the code since there may always be longer sequences with lower weight, and it converges to the free distance as the length of sequences l goes to infinity. It is defined by

$$d_l = \min_{\substack{u(D)\neq 0 \\ \deg u(D)\leq l}} w(u(D^n)g(D)). \tag{2.30}$$

Another parameter called the *column distance* corresponds to the minimum weight of nonzero paths of a certain length. It is different in the sense that paths have not necessarily remerged to the all zero sequence as it is needed for code sequences. It corresponds to the weight of the projection of code sequences onto a finite number of dimensions. Therefore, it represents a lower bound on the free distance. Although the column distance also converges to the free distance as the projection length increases, it usually converges slower to the free distance, since it does not really represent the weight of code sequences but rather the weight of truncated sequences.

Note that for catastrophic encoders, the bad mapping between the information sequences and the code sequences yields a difference between the limit of the column distance and the row distance as their order increases. For catastrophic encoders, there are unmerged long paths with low weight. Thus, the row distance of finite order l does not converge to the free distance. In order to check whether an encoder is catastrophic, we can simply check that the greatest common divisor (GCD) of all generators is 1, as proved by Massey and Sain [23]. It is important to check whether a code is catastrophic when searching for the best code, since catastrophic encoders yield a very bad probability of information bit errors.

For our problem, we are interested in the row distance since our formula gives the weight of an entire code sequence. In particular, Lemma 2.3.1 provides a new way of computing the row distance of a rate $1/n$ binary convolutional code. The following theorem relates the calculation of the row distance to Lemma 2.3.1.

Theorem 2.3.1 *Let C be a rate $1/n$ convolutional code. Then the row distance of order l of C can be computed as:*

$$d_l = \min_{\substack{u(D)\neq 0 \\ \deg u(D)\leq l}} \left(R_{0u}R_{0g} - 2\sum_{j=0}^{j=\infty} R_{1u}(j)R_{1g}(nj) + 4\sum_{j,k=0}^{\infty} R_{2u}(j,k)R_{2g}(nk,nj) - \ldots \right) \tag{2.31}$$

Proof. *Follows from Lemma 2.3.1.*

Theorem 2.3.1 gives a formula for computing the row distance of any rate $1/n$ convolutional code. In the next sections, we will see how to use this formula for constructing good convolutional codes. At the present time, most of the techniques for finding good codes are based on exhaustive search or random search such as simulated annealing. However, (2.31) can be used to direct the search towards good codes. Two algorithms are proposed in the next sections.

2.4 Algorithm I to constructing good convolutional codes

From Theorem 2.3.1, and example 2.3.1, the generator polynomial of rate 1/n convolutional codes has the following properties:

- The weight of the generator (R_{0g}) of rate 1/n convolutional codes is greater than or equal to the free distance. For good codes, it is generally equal.

- Any information sequence of weight 2 must generate a codeword with weight larger than the free distance, so for any $l > 0$,

$$2R_{0g} - 2R_{1g}(nl) \geq d_{free}. \tag{2.32}$$

Therefore, if $d_{free} = R_{0g}$, for any $l > 0$,

$$R_{1g}(nl) \leq \frac{1}{2}R_{0g}, \tag{2.33}$$

thus,

$$\max_{l} R_{1g}(nl) \leq \frac{1}{2}R_{0g}. \tag{2.34}$$

Thus, in order to construct good convolutional codes, it is necessary to find generator polynomials for which the weight of the generator is as large as possible, and the maximum first correlation coefficient is at most equal to a half of the weight. This can also be seen as generating finite binary sequences with low auto-correlation, such as pseudo-random sequences [24]. However, except for particular lengths of sequences, there is no systematic algorithm describing the construction of sequences with low auto-correlation.

Our idea is to start from the all 1's generator. The weight of the generator is maximal, but correlation coefficients are very large also. So, we replace 1 by 0 where it is advantageous in order to decrease the large correlation coefficients of the sequence. For this purpose, we can define a potential function that indicates which position is best to replace a 1 by a 0, and then iteratively replace ones by zeros in the generator until (2.34) is satisfied. The goal of the potential function is to decrease large correlation coefficients. Positions which affect large coefficients should therefore yield a large potential.

Definition 2.4.1 *Let $g^{(s)}(D)$ be the generator polynomial of a rate $1/n$ convolutional code at step s. Let $g_{(j)}^{(s+1)}(D)$ be the generator at step $s+1$, obtained by replacing a 1 by a 0 at the j^{th} position of $g^{(s)}(D)$. The potential function of $g^{(s)}(D)$ is defined by*

$$f(j) = \sum_{k=0}^{n(m+1)-1} R_{1g}^{(s)}(k)\Delta_j(k), \tag{2.35}$$

where

$$\Delta_j(k) = R_{1g}^{(s)}(k) - R_{1g_{(j)}}^{(s+1)}(k). \tag{2.36}$$

This potential function is a function of the position where a 1 can be replaced by a 0, and shows a maximum where it is advantageous in order to decrease the high correlation coefficients in $g^{(s)}(D)$. $\Delta_j(k)$ shows the difference of $R_{1g}^{(s)}(k)$ by changing 1 into 0 at the j^{th} position of the generator. We multiply a weight to this function where coefficients are large. Thus, different variations of this definition are possible such as

$$f'(j) = \sum_{k=0}^{n(m+1)-1} (R_{1g}^{(s)}(k))^2\Delta_j(k), \tag{2.37}$$

or

$$f''(j) = \sum_{k=0}^{n(m+1)-1} (R_{1g}^{(s)}(k) - \frac{1}{2}R_{0g})\Delta_j(k), \tag{2.38}$$

This leads to the following algorithm [25]:

Algorithm I:

Step 1: Let $g^{(1)}(D) = \sum_{i=0}^{n(m+1)-1} D^i$, corresponding to the all 1's generator. $s = 1$.

Step 2: Compute $f(j)$ for $j = 1, 2, \ldots, n(m+1)$.

Step 3: Let j_0 be a position for which $f(j)$ is maximum.

Step 4: $g^{(s+1)}(j_0) = 0$. $s = s + 1$.

Step 5: Compute

$$\delta = 2R_{0g} - 2 \max_{k=1,2,\ldots,n(m+1)-2} R_{1g}^{(s)}(k). \qquad (2.39)$$

Step 6: If $\delta < R_{0g}$, go to Step 2.

Step 7: If the code is catastrophic, go back to Step 2.

Step 8: Compute d_{free}. Stop.

Figure 2.2 shows the potential and the correlation functions of the generator of a rate 1/2 convolutional code with constraint length 20, after different numbers of steps in the algorithm. The curve delta represents $2R_{0g} - 2R_{1g}^{(s)}(k)$. Its maximum corresponds to δ. When δ becomes smaller than the generator weight, we stop the algorithm. In the example shown in Figure 2.2, we obtain a generator for which $R_{0g} = 23$. When we compute the free distance of the code with generators given by $g^{(1)} = (11100100101001100111 1)$ and $g^{(2)} = (11010101000110001011 1)$ and the code has only a free distance of 20.

The advantage of this method is its deterministic aspect. No search for codes is required. Unfortunately, this algorithm only guarantees a distance for inputs of weight 1 or 2. It does not insure that other inputs will lead to codewords of weight larger than the free distance. However, it is possible to use higher order correlation

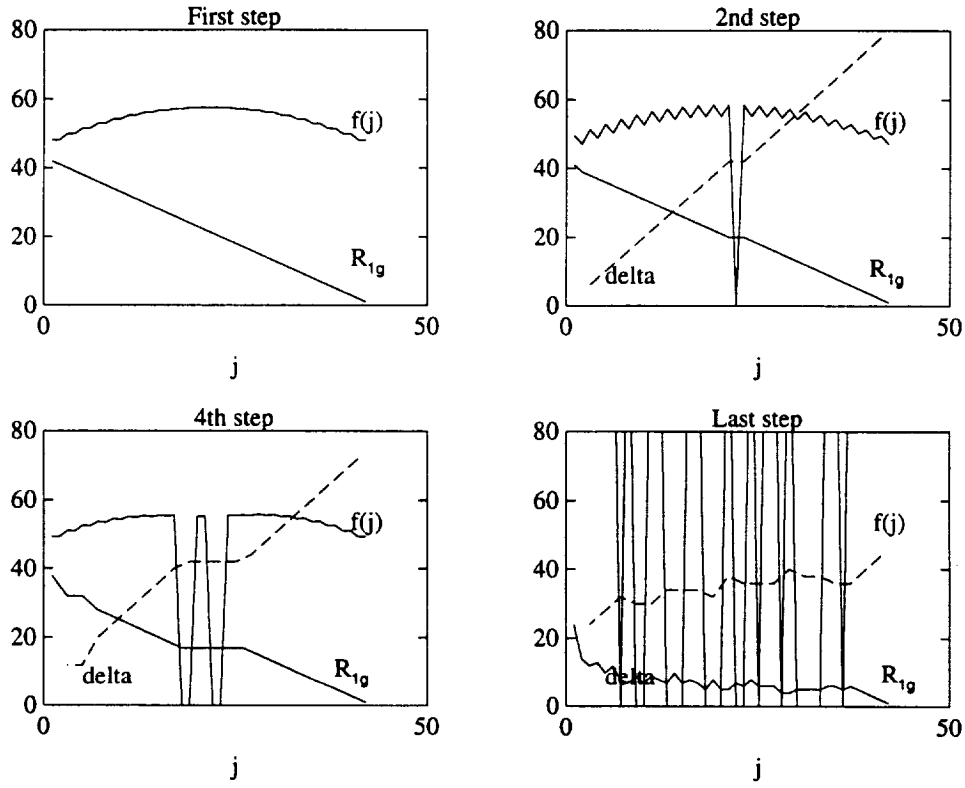

Figure 2.2: Algorithm I at different steps.

coefficients in the definition of the potential functions, and pursue the algorithm until conditions equivalent to (2.34) are satisfied for higher order correlation coefficients. It seems, however that no simple potential function leads to better codes, by pursuing the algorithm with higher order correlation coefficients. Figure 2.3 shows the free distance for rate 1/2 convolutional codes constructed by using Algorithm I.

As we can see in that figure, it is possible to find generators with much larger constraint lengths than with any algorithm involving search. However, the problem of computing the free distance for these large constraint lengths codes remains unsolved, and only an upper-bound determined by computing the row distance for information sequences of length up to 13 is given for codes with constraint length greater than 16. Thus, it has been impossible so far, to know whether this algorithm gives good codes also for large constraint length. Yet, since the codes found have good free distance for short constraint length, it strongly suggests that larger constraint lengths codes are also good.

A problem of Algorithm I is that we are not using the entire formula given in Theorem 2.3.1 but rather only the first terms, depending on the complexity of the potential function. The row distance of order 1 and 2 is often equal to the free distance for optimal codes. However, the drop of the row distance at higher order prevents these codes from reaching the free distance of optimal codes with the same constraint length. Therefore, we need to use Lemma 2.3.1 for larger orders of the row distance. Also, another more common approach with convolutional codes is to authorize some search, in order to allow us to find good codes among a larger subset of codes. By allowing search, we look for codes with a particular distance, which was

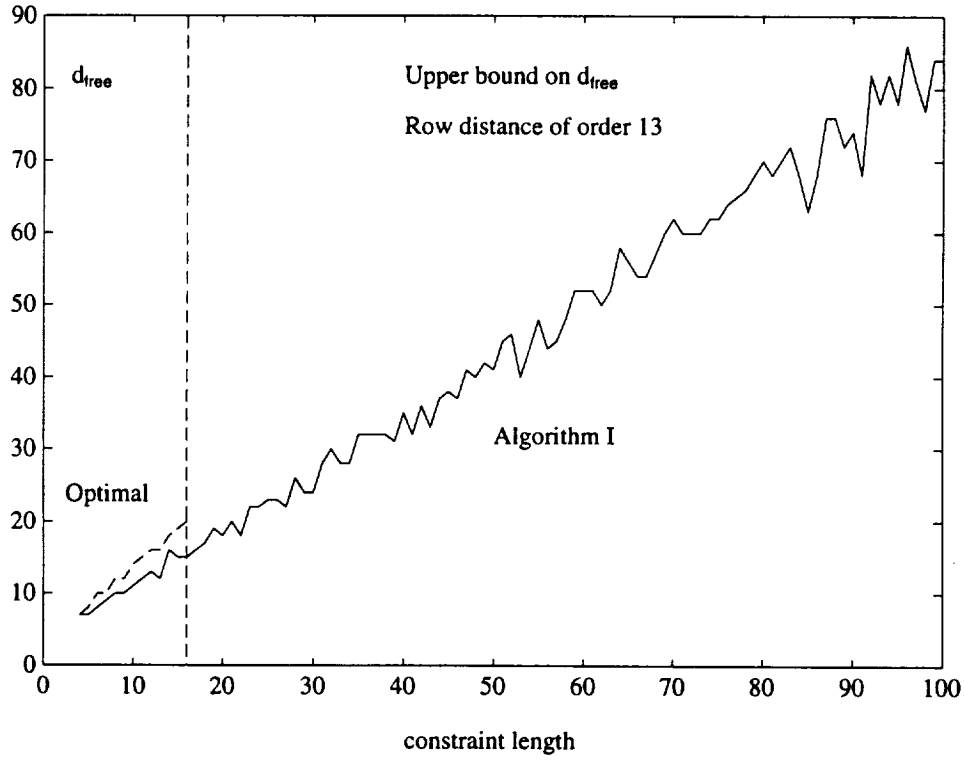

Figure 2.3: Simulation of Algorithm I for rate 1/2 codes.

not the case in Algorithm I. Thus, the goal of a search algorithm is to reduce as much as possible the number of codes that one needs to look at. This leads to another algorithm described in the next section.

2.5 Algorithm II to constructing good convolutional codes

As it was noted previously, for most of the good codes with large enough constraint length, the weight of the generator R_{0g} equals the free distance. (all rate 1/2 codes with $m \geq 4$, all rate 1/3 codes with $m \geq 2$, etc.). It was also noted that only generators presenting pseudo-random statistical properties can lead to a large free distance since they present small correlation coefficients. Therefore, it looks interesting to start from a random generator with $R_{0g} = d_{free}$, where d_{free} corresponds to an expected free distance of the code we want to construct. Then, as for a simulated annealing approach [26], we try to change successively some bits of the generator such that the free distance equates the weight of the generator.

In order to indicate which bits are advantageous to change, we can use Theorem 2.3.1 as in Algorithm I. From (2.31) indeed, it can be seen that any k^{th} correlation coefficient should be relatively small for k odd, and relatively large for k even. Yet, since all correlation coefficients are involved in (2.31), it is not obvious which coefficients are responsible for a bad distance. Therefore, it is necessary to check for the weight of the code sequences resulting from different information sequences, and in the case of a code sequence with weight less than the expected free distance, change the responsible correlation coefficients.

In order to proceed with a certain order, it is reasonable to use an algorithm to compute either the row distance or the free distance of a code to determine which information sequence leads to a 'bad' code sequence. One simple algorithm for computing the row distance consists of checking for all possible information sequences of a certain length starting with a 1.

The main idea of the algorithm is to modify the correlation coefficents in an increasing order, since the high order correlation coefficients affect codewords for information sequences of large weight only. Therefore, if a first correlation coefficient is 'bad', it will be corrected already when information sequences of weight 2 enter the encoder. So the algorithm will successively change the sequence until no 'bad' correlation coefficient leads to a low weight codeword for information sequences up to a certain length s_0. Thus, the algorithm is the following:

Algorithm II:

Step 1: Randomly choose a generator $g(D)$ such that $R_{0g} = d_{free}$, where d_{free} is the requested free distance. $s = 2$.

Step 2: Compute the weight of the code sequence generated by every information sequence starting with a 1 of length s.

Step 3: If for all sequences, the weight of the code sequences is larger than d_{free}, and $s \leq s_0$, then $s = s + 1$, go to Step 2. If $s = s_0$, go to Step 8.

Step 4: If an information sequence u leads to a codeword with

weight lower than d_{free}, then the sequence u of weight w is declared 'bad', and all the correlation coefficients of $g(D)$ involved in (2.31) are computed starting from the highest order, i.e., $w - 1$.

Step 5: The first correlation coefficient that can either be decreased (for odd order) or increased (for even order) in the list of coefficients computed in Step 4 is declared 'bad'.

Step 6: If the bad correlation coefficient needs to be decreased, a 1 is replaced by a 0 in a randomly chosen position among all the positions that affect that particular correlation coefficient. Another 0 is replaced by a 1 in a randomly chosen position among all the positions that will allow the bad correlation coefficient to decrease. $s = 2$. Go to Step 2.

Step 7: If the bad correlation coefficient needs to be increased, a 0 is replaced by a 1 in a randomly chosen position among all the positions that affect that particular correlation coefficient. Another 1 is replaced by a 0 in a randomly chosen position among all the positions that will allow the bad correlation coefficient to increase. $s = 2$. Go to Step 2.

Step 8: The code is guaranteed to have a row distance at step s_0 greater than the expected d_{free}. Stop.

As it is, algorithm II does not insure that the free distance of the code equals the one requested at the beginning, but rather that the row distance for information sequences of length up to s_0 equals the distance requested. As s_0 increases, the probability of having the free distance equal to the requested distance goes to one. The larger R_{0g} is, the less probable we will find a code that has that free distance, and the longer the search will last. In the case for which R_{0g} is greater than the free distance of any code of that constraint length, the search will not end, since no code exists with $d_{free} = R_{0g}$.

Algorithm II can easily be adapted to an algorithm that computes the free distance, which will allow us to be sure of the free distance of the constructed convolutional code, but will limit us in the constraint length of the code that we want to construct.

Table 2.1 and 2.2 show some rate 1/2 codes constructed up to constraint length $m = 27$, with expected $d_{free} = m + 3$, and number of codes searched limited to 10000, as well as one code with constraint length 50 and expected $d_{free} = 45$. The generator is given in octal form, so 70 and 64 correspond for example to 1110 and 1101. A_d represents the order of multiplicity of the paths with weight d_{free}. Figure 2.4 shows the result of the simulation of algorithm II for rate 1/2 convolutional codes with expected free distance $d_{free} = m + 3$

Algorithm II also allows to consider the multiplicity of paths with weight the free

m	g_1	g_2	d^*_{free}	A_d	number of tries
3	70	64	6	5	2
4	66	46	7	2	15
5	57	31	8	3	14
6	564	704	9	1	4
7	722	654	10	8	8
8	265	647	11	5	4
9	6474	5130	12	7	6
10	6136	5612	13	5	13
11	4713	6255	14	7	76
12	32274	63304	15	1	577
13	51262	72236	16	7	527
14	63133	71161	17	6	248
15	442734	533224	18	7	255
16	543646	655222	19	5	7830
17	744743	645047	19	1	944
18	6227254	4032474	20	6	141
19	6372112	5223626	22	1	612
20	6511203	5675311	22	10	30

$*$: d^*_{free} represents the expected free distance, and corresponds to the actual computed free distance up to $m = 16$, and to $d_0 = d_1 = \ldots = d_{13}$ for $m > 16$.

Table 2.1: Table of rate 1/2 convolutional codes constructed for constraint length $3 \leq m \leq 20$.

m	g_1	g_2	d^*_{free}	A_d	number of tries
21	77112254	56304604	23	1	529
22	25744422	45750516	24	38	3
23	42132372	57102627	24	1	8113
24	434641534	661705444	24	1	1111
25	623520432	504741376	25	17	824
26	706170627	553023125	26	19	3246
27	2743320364	4127623014	26	2	7
50	61132446441721446	40340350254005377	45	2	22

$*$: d^*_{free} represents the expected free distance, and corresponds to the actual computed free distance up to $m = 16$, and to $d_0 = d_1 = \ldots = d_{13}$ for $m > 16$.

Table 2.2: Table of rate 1/2 convolutional codes constructed for constraint length $21 \leq m \leq 27$ and $m = 50$.

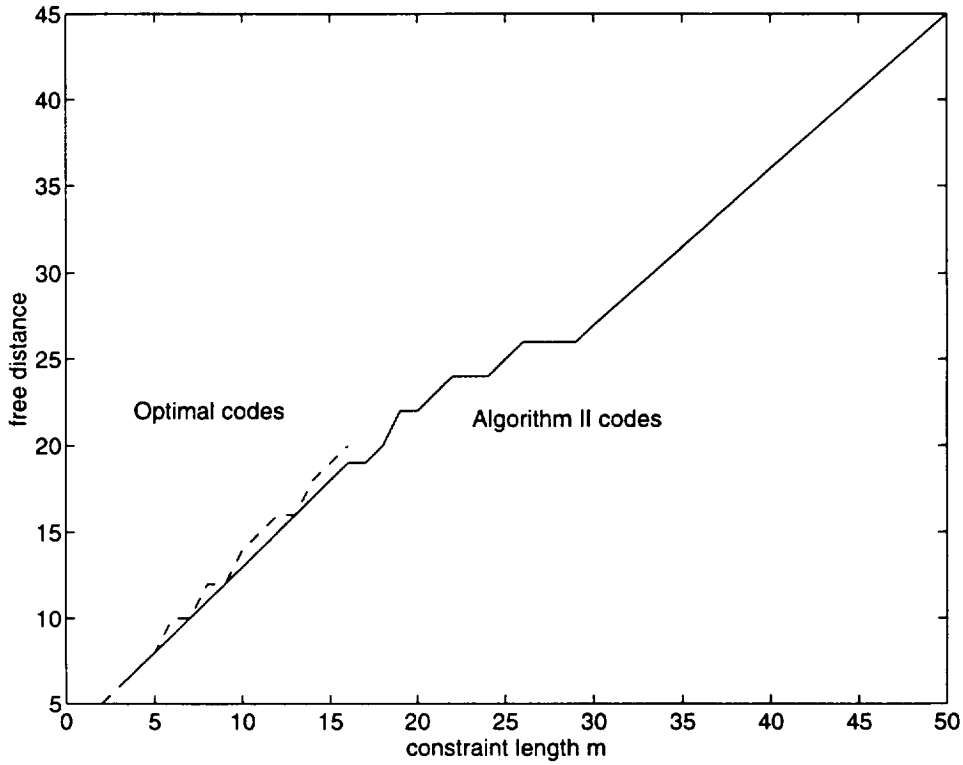

Figure 2.4: Simulation of Algorithm II for rate 1/2 codes.

distance. It is indeed possible to add a counter at Step 3 of the algorithm, such that if the multiplicity gets larger than what is expected, it declares the last sequence that leads to a codeword of weight d_{free} 'bad' and jumps to Step 4. This algorithm allows us to find a large number of codes with much larger constraint length than previously constructed codes. The main problem of the algorithm is that we are not sure whether the free distance of the constructed code equals the row distance computed at step s_0. In particular, as the constraint length increases, it becomes impossible to compute the minimum free distance of the code. A test that we can add to the algorithm in that case is to check whether the generator matrix corresponds to a catastrophic encoder. This is a simple algebraic verification which allows us to make sure that no infinite information sequence yields a low weight code sequence. However, this test does not insure whether long finite information sequence of length greater than s_0 do not lead to low weight code sequences.

2.6 Conclusion

Although binary convolutional codes can be described using algebraic notations, no algebraic construction has so far lead to good convolutional codes. In this chapter, we have presented a new approach based on a statistical method for directing the search towards good code generators. In particular, this method showed us that good binary codes are usually constructed by trying to maintain the distance given by the generator, that is, the shortest diverging and remerging path on all other paths. This concept will be used again later when we construct trellis codes over the real field.

In the following chapters, we are going to extend our construction field to real numbers. We will show indeed that the minimum free distance that one can reach with real number codes is greater than with binary codes. However, we will see that the algebraic structure of the real field has so far not lead to a generator description, and the lack of linearity between code sequences makes it even more difficult to construct good codes. In the next chapter, we are presenting bounds on the free distance of binary and real number trellis codes, in order to understand the motivation for constructing codes over the real field.

CHAPTER 3

BOUNDS ON THE DISTANCE OF BLOCK AND TRELLIS CODES

This chapter describes the general method for lower and upper bounding the minimum distance of block codes, and the free distance of trellis codes. In particular, the channel coding problem is comparable to the geometric problem of sphere packing, which studies the number of spheres of the same radius that can be packed into a given n-dimensional space. If the codewords are seen as the centers of n-dimensional non-overlapping spheres of the same radius, the minimum distance between any two codewords can be lower bounded by twice the radius of these spheres. Therefore, to prove the existence of codes with a certain minimum distance, it is sufficient to prove the existence of the corresponding sphere packings. The problem is quite different for binary codewords, because the binary n-dimensional space is a sampled version of the entire n-dimensional space, which restricts the centers of the spheres to a given set of positions in the space. In particular, the centers of the sphere are restricted to an n-dimensional hypercube, which affects the minimum distance that one can reach with a binary code.

The purpose of this chapter is to summarize the present state of the art in bounding the distance for binary and real block and trellis codes. Section 3.1 discusses the problem of sphere packing and applies it to the problem of deriving asymptotic bounds on the minimum distance of block codes. Section 3.2 reviews bounds for finite length block codes, and Section 3.3 focuses on trellis codes. Both non asymptotic and asymptotic upper bounds, as well as asymptotic lower bounds, are presented.

3.1 Sphere packing and asymptotic bounds on the minimum distance of block codes

The sphere packing problem is an old geometry problem that studies the number of spheres that can be packed into a space in n dimensions. The same problem applied to cubes is easy to solve, since there is no wasted space; however, spheres cannot be packed without losing space. The amount of space lost when packing spheres depends on the dimensionality of the space. A sphere in $\mathbb{R}^n$ with center $u = (u_1, \ldots, u_n)$ and radius r consists of all the points $x = (x_1, \ldots, x_n)$ that satisfy

$$(x_1 - u_1)^2 + (x_2 - u_2)^2 + \ldots + (x_n - u_n)^2 = r^2. \tag{3.1}$$

A sphere in one dimension is simply a segment as shown in Figure 3.1, and therefore no space is lost over the real line. In dimensions greater or equal to two, some space is lost. In order to evaluate the amount of space lost between the spheres, the concept of packing density is introduced. The density Δ of a packing corresponds to the proportion the volume used by non overlapping spheres of the same radius situated in a larger sphere to the volume of that sphere. As the volume of the large sphere increases, this ratio converges to a limit called *packing density.* For each dimension

n, there is an infinite number of ways to pack spheres into the space. However, one type of packing called lattice packing is particularly interesting, because it provides a simple evaluation of the density. A lattice packing in n dimensions is defined by n centers $v_1, \ldots, v_n$ such that any center v of the lattice can be expressed by the sum

$$v = \sum_{i=1}^{n} k_i v_i, \tag{3.2}$$

where k_i, for $i = 1, \ldots, n$, are integers. The set of n points in n dimensions $(v_1, \ldots, v_n)$ forms a basis for the lattice, and if the coordinates of these basis vectors in an m-dimensional orthonormal basis with $m \geq n$ are

$$\begin{aligned} v_1 &= (v_{11}, v_{12}, \ldots, v_{1m}), \\ v_2 &= (v_{21}, v_{22}, \ldots, v_{2m}), \\ &\vdots \\ v_n &= (v_{n1}, v_{n2}, \ldots, v_{nm}), \end{aligned} \tag{3.3}$$

then the generator matrix of the lattice is defined by

$$M = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1m} \\ v_{21} & v_{22} & \cdots & v_{2m} \\ \vdots & & & \vdots \\ v_{n1} & v_{n2} & \cdots & v_{nm} \end{bmatrix}. \tag{3.4}$$

Thus, the density can be computed by

$$\Delta = \frac{V_n r^n}{(det(MM^T))^{1/2}}, \tag{3.5}$$

where

$$V_n = \frac{\pi^{n/2}}{\Gamma(n/2) + 1}. \tag{3.6}$$

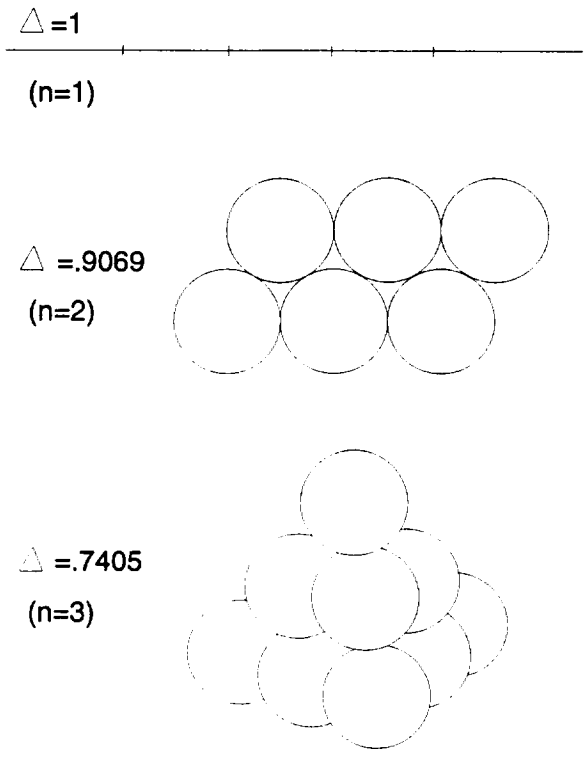

Figure 3.1: Best packings in 1, 2 and 3 dimensions.

Although lattices are particular types of packing, they are simpler to construct than other packings, provide a formula for computing the density, and, most importantly, lead to bounds on the density of all types of packings in n dimensions. That is, a lower bound on the density of the best lattices also represents a lower bound on the density of any sphere packing. Also, because of the linear structure of most binary codes, codewords form lattices, and upper bounds on the density of lattices can be used to upper bound the minimum distance of codes. The following section presents lower and upper bounds on the density of lattices, which will be used later to derive lower and upper bounds on the distance of codes.

3.1.1 Bounds on the density of lattices

It was shown by Minkowski [27] that there exist lattices with density satisfying

$$\Delta \geq \frac{\mathcal{Z}(n)}{2^{n-1}}, \tag{3.7}$$

where $\mathcal{Z}(n) = \sum_{k=1}^{\infty} k^{-n}$. This means in particular, that as n approaches infinity,

$$\log_2 \Delta \geq -n + 1. \tag{3.8}$$

However, Minkowski's proof is not constructive and no method is known to construct lattices with such a good density.

The first upper bound on lattice densities was found by Rogers [27], who showed that

$$\Delta \leq \sigma_n, \tag{3.9}$$

where σ_n is defined by the ratio of the volume of the part of a regular n-dimensional simplex of edge length 2 covered by spheres of radius 1 centered at the vertices of the simplex to the total volume of the simplex. Table 3.1 gives σ_n for the first 10 values of n [28]. We will use these values to upper bound the minimum distance of block codes in section 3.2. Also, in the limit as n goes to infinity, this proves that

$$\log_2 \Delta \leq -\frac{n}{2}. \tag{3.10}$$

However, recently, Kabatiansky and Levenshtein [29] used linear programming method to prove that the maximal number $A(n, \theta)$ of points situated on an n-dimensional sphere separated by at least an angle θ verifies for large n and $0 < \theta < \pi / 2$,

$$\frac{1}{n} \log_2 A(n, \theta) \leq \frac{1 + \sin\theta}{2 \sin\theta} \log_2 \frac{1 + \sin\theta}{2 \sin\theta} - \frac{1 - \sin\theta}{2 \sin\theta} \log_2 \frac{1 - \sin\theta}{2 \sin\theta}, \tag{3.11}$$

dimension n	Upper bound on the density Δ
1	1.000
2	.9069
3	.7796
4	.6478
5	.5257
6	.4192
7	.3298
8	.2568
9	.1981
10	.1518

Table 3.1: Upper bound on the density of sphere packings for $n \leq 10$.

which for $\theta < 63^o$ simplifies into

$$\frac{1}{n}\log_2 A(n,\theta) \leq -\frac{1}{2}\log_2(1-\cos\theta) - 0.099. \tag{3.12}$$

It can be shown that the packing density can be upper bounded by

$$\Delta \leq (\sin\frac{\theta}{2})^n A(n+1,\theta), \tag{3.13}$$

for $0 < \theta \leq \pi$, thus yielding for $\theta = 63^o$,

$$\log_2 \Delta \leq -0.599n, \tag{3.14}$$

as n goes to infinity.

In the next section, the relation between the sphere packing problem and error correcting codes will be studied. This will allow us to first derive lower and upper bounds on the minimum distance of block codes, and then use those bounds to develop bounds for trellis codes. Although the bounds are not always tight enough to indicate the distance of existing codes, they will provide a comparison of the possible performance of real and binary codes.

3.1.2 Relation between code distance and packing density

Let us consider a large sphere corresponding to the maximum power nP that one wants to transmit per n-dimensional codeword. Figure 3.2 shows such a sphere for 2 dimensions. Let M be the number of n-dimensional spheres of small radius r that one can put in the large sphere. From the definition of the sphere packing density, in the limit as M approaches infinity,

$$MV_n r^n = \Delta V_n (nP)^{n/2}. \tag{3.15}$$

Since the minimum squared Euclidean distance d_E^2 between the centers of the small spheres is equal to

$$d_E^2 = (2r)^2, \tag{3.16}$$

then

$$\lim_{M\to\infty} d_E^2 = 4nP(\frac{\Delta}{M})^{2/n}. \tag{3.17}$$

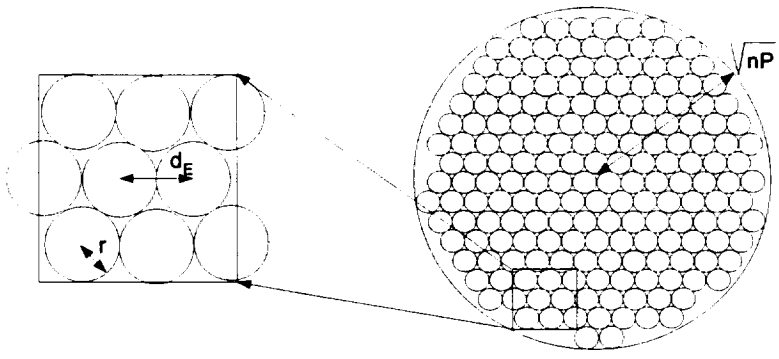

Figure 3.2: Distance between codewords packed in a maximum energy sphere.

3.1.3 Asymptotic bounds on the minimum distance of block codes

Real number codes

In (3.17), the distance corresponds to *block* codes of dimension n. Using this relationship, an asymptotic upper bound on the Euclidean distance of block codes can be derived by using (3.14) which yields for R not too small (M large)

$$d_E^2 \leq 4nP(\frac{2^{-0.599n}}{2^{nR}})^{2/n}, \tag{3.18}$$

where the rate $R = \frac{1}{n}\log_2 M$. However, for R small, M is to small to use the approximation (3.17), and it is necessary to directly upper bound the minimum squared Euclidean distance by using (3.11) and noting that the distance between two points situated on an n-dimensional sphere and separated by an angle θ is $d_E^2 = 4nP\sin^2(\frac{\theta}{2})$. Thus, by inverting (3.11), we obtain an upper bound on θ as a function of R, which yields an upper bound on d_E^2. For $\theta < 63°$, that is for $R > .34$, this simplifies into (3.18).

Theorem 3.1.1 *The minimum squared Euclidean distance d_E^2 of a real number block code with rate $R > .34$ and large n is upper bounded by*

$$d_E^2 \leq \frac{nP}{2^{0.198}}2^{1-2R}. \tag{3.19}$$

For $R < .34$, θ is upper bounded by

$$R \leq \frac{1+\sin\theta}{2\sin\theta}\log_2\frac{1+\sin\theta}{2\sin\theta} - \frac{1-\sin\theta}{2\sin\theta}\log_2\frac{1-\sin\theta}{2\sin\theta}, \tag{3.20}$$

which yields an upper bound on $d_E^2 = 4nP\sin^2(\frac{\theta}{2})$.

An asymptotic lower bound can also be derived for R large (M large) by using (3.8). Indeed, in the limit as M goes to infinity $d_E^2 = 4nP(\frac{\Delta}{M})^{2/n}$. Thus, since $\Delta \geq 2^{-n}$ as n goes to infinity,

$$d_E^2 \geq 4nP(\frac{2^{-n}}{2^{nR}})^{2/n}. \tag{3.21}$$

However, the Gilbert-Varshamov asymptotic lower bound on binary block codes [30] is given by

$$R \geq 1 - H_2(\frac{d_H}{n}), \tag{3.22}$$

for all n, where d_H is the Hamming distance and

$$H_2(x) = -x\log_2 x - (1-x)\log_2(1-x) \tag{3.23}$$

is the binary entropy function. Since this lower bound is valid for binary block codes, it is also valid for real number codes with $d_E^2 = 4Pd_H$. This yields the following theorem.

Theorem 3.1.2 *The minimum squared Euclidean distance d_E^2 of a real number block code is lower bounded by*

$$d_E^2 \geq \max(4nPH_2^{-1}(1-R), nP2^{-2R}) \tag{3.24}$$

Proof. *Follows from (3.21) and (3.22).*

Note that for $R < .41$, the Gilbert-Varshamov bound is the best bound, for $R > .41$, (3.21) is the best bound.

Binary block codes

Binary codes are restricted to a sampled version of the entire Euclidean space. For that reason, it is necessary to use algebraic arguments to derive tight bounds on the Hamming distance of the code, which is proportional to the squared euclidean distance after modulation. The main problem for binary block codes is determining the function $A(n,d)$, which is defined as the maximal number of binary vectors of length n that can be found with the property that any of two of the vectors differ in at least d places. In particular, the rate of the code R is related to $A(n,d)$ by

$$R = \frac{1}{n}\log_2 A(n,d). \tag{3.25}$$

Mc Eliece, Rodemich, Rumsey and Welch [31] proved that for $0 \leq d/n < 1/2$,

$$R(n,d) \leq \min_{0\leq u\leq 1-2d/n}\{1 + h(u^2) - h(u^2 + \frac{2du}{n} + \frac{2d}{n})\}, \tag{3.26}$$

in the limit as n goes to infinity, where $h(x) = H_2((1-\sqrt{1-x})/2)$. In the range $.273 < d/n \leq .5$, the minimum in (3.26) is attained at $u = 1 - 2d/n$, so the bound simplifies to

$$R \leq H_2\left(\frac{1}{2} - \left\{\frac{d}{n}\left(1-\frac{d}{n}\right)\right\}^{1/2}\right). \tag{3.27}$$

In order to compare this bound with the bounds for real codes, assuming we send binary codewords using BPSK, $d_E^2 = 4Pd$, thus inverting (3.27), we obtain the following theorem.

Theorem 3.1.3 *The minimum squared Euclidean distance of a binary block code transmitted using BPSK modulation is upper bounded by*

$$d_E^2 \leq 2nP(1 - \sqrt{1 - 4\left(\frac{1}{2} - H_2^{-1}(R)\right)^2}). \tag{3.28}$$

The best lower bound on R comes from the Gilbert-Varshamov bound, thus the following theorem gives an asymptotic lower bound on the minimum squared Euclidean distance of binary block codes.

Theorem 3.1.4 *The minimum squared Euclidean distance of a binary block code transmitted using BPSK modulation is lower bounded by*

$$d_E^2 \geq 4nPH_2^{-1}(1-R) \tag{3.29}$$

The asymptotic lower and upper bounds on the minimum squared euclidean distance for binary and real number block codes described by theorems 3.1.1, 3.1.2, 3.1.3, and 3.1.4 are represented in figure 3.3. Note that the upper bounds for binary and real number codes are almost equal for $R < .2$. This graph shows that for $R > .2$, the upper bound for binary codes starts being lower than the upper bound for real number codes, for $R > .4$, the lower bound on real number codes starts being greater than the lower bound for binary codes, and for $R > .75$, there exists real number codes that reach a greater distance than any binary code, since the lower bound for real number codes crosses the upper bound for binary codes. This shows that for rates greater than .75, real number codes are asymptotically better than binary codes of the same rate. Finally, for $R > 1$, binary codes cannot provide any euclidean distance, since no redundancy is added to the information sequence, whereas real number codes can still provide some distance.

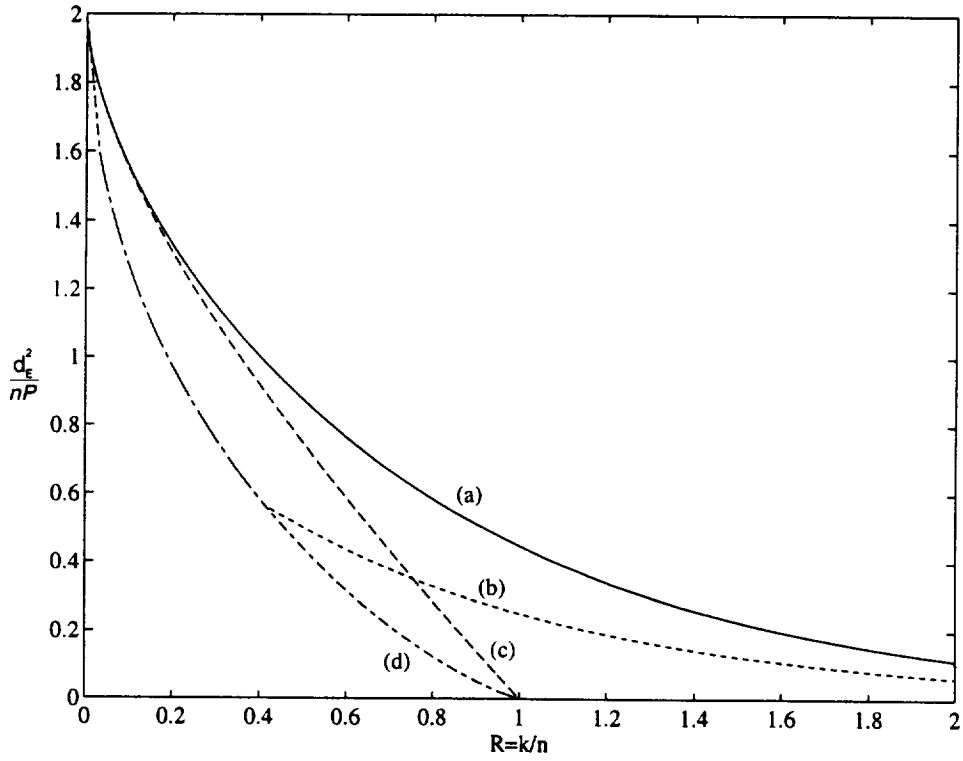

Figure 3.3: Asymptotic lower and upper bounds on the minimum squared Euclidean distance of binary and real block codes: (a): Real codes upper bound, (b): Real codes lower bound, (c): Binary codes upper bound, (d): Binary codes lower bound

3.2 Non asymptotic bounds for block codes

The main problem which arises when trying to compute non asymptotic bounds on the distance of block codes is to estimate the best sphere packing for finite sets of points, which is not as simple as the asymptotic problem. In particular, for block codes of rate lower than 1, we can use the simplex bound which evaluates the distance between the points of a simplex structure in n dimensions. This gives a good upper bound as long as the number of codewords is smaller than the number of points in an n-dimensional hypercube, that is, as long as $M < 2^n$, *i.e.* $k < n$. This bound on the number $A(n, d)$ introduced in the previous section was derived by Plotkin [32] and is given by

$$A(n, d) \leq 2 \left\lfloor \frac{d}{2d - n} \right\rfloor \tag{3.30}$$

for $n < 2d$, and $d = d_E^2/4$. This bound is equivalent to the regular simplex bound or Roger's bound on lattice density seen in the previous section

$$d_E^2 \leq \frac{2nPM}{M - 1}. \tag{3.31}$$

Therefore, this bound applies to codewords constructed on the real number field, but becomes too weak for rates greater than 1.

For binary codes, which always have rates lower than 1, this theorem can be strengthened by the Hamming bound [32]

$$A(n, 2\delta + 1) \left(1 + \binom{n}{1} + \ldots + \binom{n}{\delta} \right) \leq 2^n, \tag{3.32}$$

where $d = 2\delta + 1$. This be tightened by using the Johnson bound [32]

$$A(n,2\delta+1)\left\{1+\binom{n}{1}+\ldots+\binom{n}{\delta}+\frac{\binom{n}{\delta}-\left(\frac{n-\delta}{\delta+1}-\left\lfloor\frac{n-\delta}{\delta+1}\right\rfloor\right)}{\left\lfloor\frac{n}{\delta+1}\right\rfloor}\right\}\leq 2^n. \quad (3.33)$$

Moreover, if the binary code is linear, Griesmer [33] derived a bound using the generator matrix of the binary block codes

$$\sum_{i=0}^{k-1}\lceil\frac{d}{2^i}\rceil\leq n. \quad (3.34)$$

However, for rates greater than 1, the number of codewords that must be constructed in $\mathbb{R}^n$ is strictly greater than the number of vertices in the n-dimensional hypercube, and the simplex bound becomes weak. It is therefore necessary to put codewords on a larger space than the n dimensional hypercube. In that case, it is worthy using the sphere packing problem which estimates the maximum number of codewords separated by a squared euclidean distance d_E^2 in an n-dimensional sphere. Since the average energy of the block code is smaller than the maximum energy used to send an n-dimensional symbol, we can use (3.17) to upper bound the distance by

$$d_E^2\leq 4nP(\frac{\Delta}{M})^{2/n}. \quad (3.35)$$

The density Δ can itself be upper bounded by Roger's bound seen in (3.9). Table 3.2 gives the upper bound on the normalized distance per dimension computed from (3.35) for code rates greater than 1, and from the minimum between the Hamming bound, the Griesmer bound and the Johnson bound for rates lower than 1. Note however, that for rates greater or equal to 1, table 3.2 gives an upper bound on the distance normalized to the maximum energy needed to transmit any codeword. This

	k									
n	1	2	3	4	5	6	7	8	9	10
1	4.000	0.250	0.062	0.016	0.004	0.001	0.000	0.000	0.000	0.000
2	4.000	2.000	0.453	0.227	0.113	0.057	0.028	0.014	0.007	0.004
3	4.000	2.667	1.333	0.534	0.336	0.212	0.133	0.084	0.053	0.033
4	4.000	2.000	2.000	1.000	0.569	0.402	0.285	0.201	0.142	0.101
5	4.000	3.200	1.600	1.600	0.800	0.586	0.444	0.337	0.255	0.193
6	4.000	2.667	2.667	1.333	1.333	0.748	0.594	0.471	0.374	0.297
7	4.000	3.429	2.286	1.714	1.143	1.143	0.728	0.598	0.490	0.402
8	4.000	3.000	2.000	2.000	2.000	1.007	1.000	0.712	0.599	0.503
9	4.000	2.667	2.667	1.778	1.778	1.333	0.950	0.889	0.698	0.598
10	4.000	3.200	2.400	2.400	1.600	1.600	1.200	0.905	0.800	0.686
	k									
n	11	12	13	14	15	16	17	18	19	20
1	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.002	0.001	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
3	0.021	0.013	0.008	0.005	0.003	0.002	0.001	0.001	0.001	0.000
4	0.071	0.050	0.036	0.025	0.018	0.013	0.009	0.006	0.004	0.003
5	0.146	0.111	0.084	0.064	0.048	0.037	0.028	0.021	0.016	0.012
6	0.236	0.187	0.149	0.118	0.094	0.074	0.059	0.047	0.037	0.029
7	0.330	0.271	0.222	0.182	0.149	0.123	0.101	0.082	0.068	0.055
8	0.423	0.356	0.299	0.252	0.212	0.178	0.150	0.126	0.106	0.089
9	0.513	0.440	0.377	0.323	0.277	0.237	0.204	0.174	0.150	0.128
10	0.597	0.520	0.453	0.394	0.343	0.299	0.260	0.226	0.197	0.171

Table 3.2: Upper bound on d_E^2/nP for (n, k) block codes

energy is greater than the average energy over all possible codewords. Therefore, it is possible to construct some real block codes for which the euclidean distance divided by the average energy is greater than the value indicated in the table, even though it is an upper bound.

These bounds are shown in Figure 3.4 for rate $k/10$ block codes with $1 \leq k \leq 20$. Note that for rates greater than 1, only real number codes can reach a nonzero distance. For rates lower than 1, the real number codes can reach a better distance than binary codes for almost all rates. These results using non asymptotic bounds are

consistent with the results given by the asymptotic bounds in the previous section. In the next section, we will use these non-asymptotic bounds on the minimum distance of block codes to derive bounds on the free distance of binary and real trellis codes.

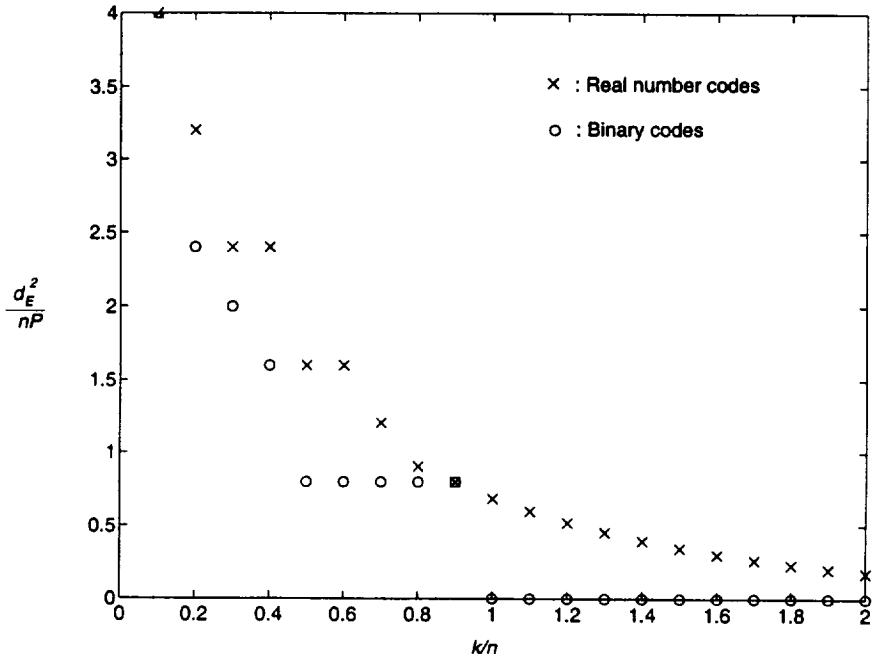

Figure 3.4: Upper bounds on the minimum squared Euclidean distance of binary and real block codes of finite dimension $n = 10$ sent with maximum energy nP.

3.3 Trellis codes

Upper bounds for trellis codes are based on upper bounds for block codes, since trellis codes can be seen as a set of code sequences of length varying from $\lceil \frac{m}{k} \rceil + 1$ to infinity. Thus, all code sequences of a certain length constitute block codes for which asymptotic and non asymptotic upper bounds are known. Therefore, the minimal upper bound over all possible sequence lengths constitutes an upper bound for trellis

codes. For trellis codes, we first recall the non asymptotic bounds on the free distance for real and binary block codes, and then derive asymptotic upper bounds based on Calderbank, Mazo, and Wei's method [34].

3.3.1 Non asymptotic upper bounds for trellis codes

The first non asymptotic bound for real number codes was given by Calderbank, Mazo, and Wei [34], who based their upper bound on the Plotkin upper bound for block codes, seen in (3.31). They noted that every path of length l through the trellis determines a vector of dimension nl. Supposing that A and B are two trellis states, the number of paths from A to B of length l is $M_{AB}(l) = 2^{kl-m}$. Assuming nP is the average energy to transmit an n-dimensional symbol, there exists a pair of states A and B such that the energy to transmit any of the $M_{AB}(l)$ codewords is lower than lnP. (3.31) allows us to upper bound the squared free Euclidean distance between two nl-dimensional codewords taken among M nl-dimensional codewords packed in a sphere of energy lnP. Thus,

$$d_E^2(l) \leq \frac{2lnPM}{M-1}. \tag{3.36}$$

Taking the minimum over all possible paths of length l for l varying from $\lceil \frac{m}{k} \rceil + 1$ to infinity, we obtain an upper bound on the squared free Euclidean distance of the trellis code. Therefore, letting $t = l - \lceil \frac{m}{k} \rceil$,

$$d_{free}^2 \leq 2nP \min_{t \geq 1} \left[\frac{2^{tk}}{2^{tk}-1} \left(\left\lceil \frac{m}{k} \right\rceil + t \right) \right]. \tag{3.37}$$

In particular, this generalizes Heller's bound [35, 36] for binary linear convolutional codes for which $P = 1$ to transmit one bit, and assuming we are sending binary digits

using BPSK, the Hamming distance equals 1/4 of the squared Euclidean distance (see Section 1.2),

$$d_{free_H} \leq \frac{n}{2} \min_{t \geq 1} \left[\frac{2^{tk}}{2^{tk}-1} \left(\left\lceil \frac{m}{k} \right\rceil + t \right) \right]. \tag{3.38}$$

Note however that the same problem that exists for block codes occurs for trellis codes when the rate is lower than 1, since the Plotkin or simplex bound is weak when the rate of the block code is greater than 1. Although convolutional codes always have $2^{kl-m} \leq 2^{nl}$, it is not the case for real number trellis codes with rates lower than 1. This was first noted by Pottie and Taylor [37] who used tighter bounds as those derived in section 3.2 for small block codes. As expected, their bound is much tighter for codes such as Ungerboeck's trellis codes [3] when k increases. We will use their bound in the next chapters to compare our constructions with the upper bound.

As in section 3.2, it is also possible to tighten the bound for linear binary convolutional codes by using the Griesmer bound. Using (3.34) and considering all paths that diverge and remerge from the all zero state after $\lceil \frac{m}{k} \rceil + i, i = 1, 2, \ldots$, the minimum squared Euclidean distance must verify

$$\sum_{i=0}^{kt-1} \left\lceil \frac{d_{free_H}}{2^i} \right\rceil \leq (\left\lceil \frac{m}{k} \right\rceil + t)n, \tag{3.39}$$

where $d_{free_H} = d^2_{free}/4$.

It has been shown that it is possible to apply all upper bounds derived on block codes to trellis codes. In particular, in [37], Pottie and Taylor use sphere packing argument to tighten the upper bound for trellis codes constructed on certain constellations, and in [38], Calderbank and Pottie derive upper bounds based on the Johnson bound (see (3.33)). For asymptotic upper bounds, Costello derived bounds

for binary convolutional codes [39]. The first technique to derive asymptotic upper bounds for real number trellis codes was discovered by Calderbank, Mazo and Wei [34]. In the next section, we give the result of their bound, show that their result is only an approximation of the correct bound for certain code rates, thus yielding a slightly less tight bound for certain rates. We also use this bound to derive a new upper bound for binary convolutional codes, which is more dependent on the rate than Costello's bound.

3.3.2 Asymptotic upper bounds for trellis codes

Real Trellis Codes

Calderbank, Mazo, and Wei derived an asymptotic upper bound on real trellis codes by bounding it with a sequence of suitably chosen block codes for which upper bounds can be derived using the most recent results in Sphere Packing on the packing density (see Section 3.1.1, 3.11 and 3.12). In particular, they prove that if there exists a trellis code with distance d_{free} and average transmitted signal power nP per symbol, then for any $\alpha > 0$ with $(1+\alpha)m/k$ an integer, there exists a Euclidean block code with average power per dimension lower than P, rate $R \geq (\alpha/(1+\alpha))k/n$, dimensionality $n' = (1+\alpha)mn/k$, and minimum distance $d_{min} \geq d_{free}^2/n'$.

This allows them to upper bound the free distance of a trellis code by an upper bound on the distance between codewords of a n' dimensional block code of rate $R = (\alpha/(1+\alpha))k/n$. This upper bound is given by Theorem 3.1.1. So, if $R =$

$(\alpha/(1+\alpha))k/n > .34$, then

$$d_{min}^2 \leq \frac{(1+\alpha)mn/kP}{2^{0.198}} 2^{1-2\alpha k/(1+\alpha)n}, \tag{3.40}$$

so

$$d_{free}^2 \leq \min_{\alpha > \frac{.34n}{k-.34n}} \left[\frac{2nmP}{2^{0.198}k} \cdot \frac{(1+\alpha)}{4^{(\alpha k/(1+\alpha)n)}} \right]. \tag{3.41}$$

The right is minimized by choosing $\alpha = ((k/n)\ln 4) - 1$. Hence, if $k/n \geq (1 + .34\ln 4)/(\ln 4) > 1.06$,

$$d_{free}^2 \leq \frac{6.57mP}{4^{k/n}}, \tag{3.42}$$

For $k/n < 1.06$, we have to use (3.20) in Theorem 3.1.1. This can be done by letting α vary, and find θ_α such that

$$\frac{\alpha k}{(1+\alpha)n} = \frac{1+\sin\theta_\alpha}{2\sin\theta_\alpha} \log_2 \frac{1+\sin\theta_\alpha}{2\sin\theta_\alpha} - \frac{1-\sin\theta_\alpha}{2\sin\theta_\alpha} \log_2 \frac{1-\sin\theta_\alpha}{2\sin\theta_\alpha}. \tag{3.43}$$

The upper bound on the distance is then given by

$$d_{free}^2 \leq \min_{\alpha>0} 4(1+\alpha)m\frac{n}{k}P\sin^2\left(\frac{\theta_\alpha}{2}\right). \tag{3.44}$$

Note that the upper bound in [34] was derived using (3.41) even for rates $k/n < 1.06$. Their result is therefore only an approximation of the correct bound for this range of rates. This is particularly important when comparing the bound we just derived with bounds on binary trellis codes, since the upper bound on the free distance for binary trellis codes would be greater than the upper bound on the free distance for real trellis codes, which is not the case when using the exact calculation of the upper bound as it was just executed. In the next subsection, we calculate a new upper bound on the free distance of binary trellis codes based on the same approach but

using upper bounds on binary block codes. This allows us to find a tighter upper bound on the free distance than the one derived by Costello in [39].

Binary convolutional codes

For binary convolutional codes, we can use the asymptotic upper bound on binary block codes in Theorem 3.1.3 and apply it to the proof of Calderbank, Mazo and Wei. Thus, an upper bound on the minimum free squared Euclidean distance of binary convolutional codes is given by

$$d^2_{free} \leq \min_{\alpha>0} 2(1+\alpha)m\frac{n}{k}P(1-\sqrt{1-4\left(\frac{1}{2}-H_2^{-1}(\frac{\alpha k}{(1+\alpha)n})\right)^2}). \tag{3.45}$$

This upper bound and the exact calculation of Calderbank *et.al.*'s asymptotic upper bound on the minimum free square Euclidean distance of real number trellis code are plotted in Figure 3.5.

Here also, we observe that the asymptotic upper bound on the minimum squared free distance of binary convolutional codes is tighter than the upper bound for real number codes, especially for rates greater than 1. However, note that for rates lower than 1, the two bounds are equal, which suggests that as the constraint length increases, the free distance obtained with binary and real number convolutional codes may be the same. We will see in chapter 5 that the real number codes indeed seem to perform just as good as binary convolutional codes for large constraint length.

Note that Costello's upper bound on the minimum free distance of binary convolutional codes [39] is given by

$$\lim_{m\to\infty} d_{free} < 2nPm \tag{3.46}$$

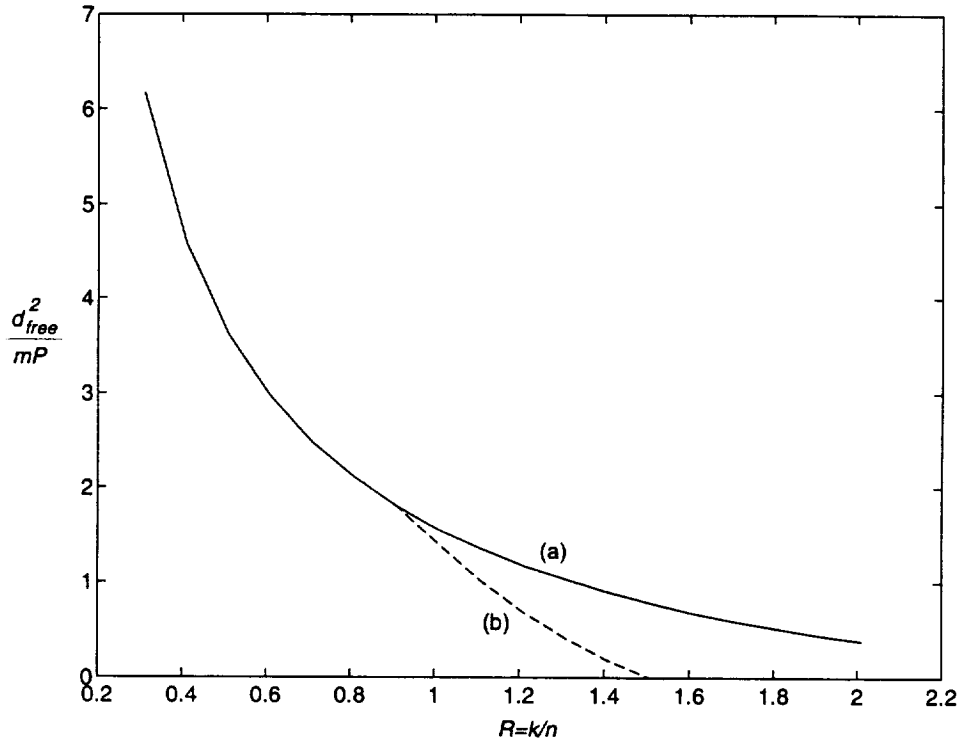

Figure 3.5: Asymptotic upper bounds on the minimum free squared Euclidean distance of (a) real trellis codes and (b) binary trellis trellis codes.

which depends on n rather than on the rate R. As R decreases, n is forced to go to infinity, which corresponds to our bound. For other rates, our bound is more specific than Costello's bound.

In the next section, we will investigate lower bounds on the free distance of trellis codes. These lower bounds are more difficult to establish than upper bounds, since they prove the existence of codes that can reach the distance given by the bound. This requires Gilbert-type [30] counting argument which usually lead to weak lower bounds instead of the distance achievable by codes.

3.3.3 Asymptotic lower bounds on the free distance of trellis codes

Costello [39] first derived lower bounds on the minimum free Hamming distance of binary convolutional codes. More recently, Rouanne and Costello [40] have calculated a bound for real trellis codes based on Ungerboeck's construction [3]. However, Chao and Chiu [41] have written a comment on that last paper explaining that Rouanne and Costello's bound uses the linear structure of binary convolutional codes allowing them to compute the free distance from the all zero sequence, while real number trellis codes do not necessary present such a property. The new bound that Chao and Chiu derived is unfortunately much too weak to be useful in practice. We will first present Rouanne and Costello's bound, which also gives a bound on binary convolutional codes, and then present Chao and Chiu's comments on the bound calculation.

Lower bound on the minimum free distance of trellis codes

The usual start for the derivation of a lower bound on the distance reached by a code is the following

$$\Pr_{c\in C}(d_{free}(c) \leq d) < 1 \Rightarrow \exists c_0 \in C \text{ such that } d_{free}(c_0) > d. \tag{3.47}$$

In order to bound the distance, it is therefore sufficient to derive an upper bound on $\Pr_{c\in C}(d_{free}(c) \leq d)$, and force this upper bound to 1, in order to find the free distance d which can surely be reached by a code. First, if there is a code which has distance lower than d, then there exists two distinct paths v_0 and v_0' such that the euclidean distance between them is less than d. This can be expressed as

$$e^{\alpha d^2} e^{-\alpha d_E^2(v_0,v_0')} \geq 1, \tag{3.48}$$

where α can be any positive real number. We can upper bound the left side of (3.48) by summing over all v_0', thus

$$e^{\alpha d^2} \sum_{v'} e^{-\alpha d_E^2(v_0,v')} \geq 1, \tag{3.49}$$

which can be raised to a power $\rho \geq 0$, yielding

$$e^{\alpha\rho d^2} \left(\sum_{v'} e^{-\alpha d_E^2(v_0,v')} \right)^{\rho} \geq 1 \tag{3.50}$$

for all α and ρ greater than 0. By summing over all correct paths v_0,

$$e^{\alpha\rho d^2} T_{\alpha,\rho}(c) \geq 1, \tag{3.51}$$

where

$$T_{\alpha,\rho}(c) \triangleq \sum_{v\in C} \left(\sum_{v'} e^{-\alpha d_E^2(v,v')} \right)^{\rho}. \tag{3.52}$$

Therefore, by averaging over all possible codes,

$$\Pr_{c\in C}(d_{free}(c) \leq d) \leq e^{\alpha\rho d^2} \sum_{c\in C} p(c) T_{\alpha,\rho}(c), \tag{3.53}$$

for all α and ρ. The next step consists in expanding the previous sum and switching the sum over the codes in C with the sum over all possible code sequences v in each code, as it was first done by Shannon [2] in his derivation of the random coding bound. This leads to

$$\sum_{c\in C} p(c) T_{\alpha,\rho}(c) \leq \sum_{v} p(v) \left[\sum_{v'} \prod_{t=1}^{\infty} p(v'_t/v_t) e^{-\alpha d_E^2(v_t, v'_t)} \right]^{\rho}, \tag{3.54}$$

where a sequence v is decomposed into the symbols v_t sent at time t. By considering the topology of the trellis structure, it is then possible to average over all possible signal labelings on paths with the same topology. This step called *configuration counting* by Forney [42] yields

$$\Pr_{c\in C}(d_{free}(c) \leq d) \leq e^{\alpha\rho d^2} \sum_{\tau=\nu+1}^{\infty} 2^{k\tau\rho - k\nu\rho} e^{-\tau\rho E(\alpha,\rho)}, \tag{3.55}$$

where

$$E(\alpha,\rho) \triangleq -\ln \left[\sum_{s\in S} p(s) \left(\sum_{s'\in S} p(s') e^{-\alpha d_E^2(s,s')} \right)^{\rho} \right]^{1/\rho}, \tag{3.56}$$

where s and s' are symbols of the signal constellation S, and $p(s)$ the probability of transmitting s.

By forcing the right term of (3.55) to be strictly less than 1, Rouanne and Costello obtain a lower bound on the free distance of trellis codes

$$d_{free}^2 > \max_{\substack{E(\alpha,\rho) > k\ln 2 \\ \alpha \geq 0 \\ 1 \geq \rho \geq 0 \\ p(s)}} \left(\nu \frac{E(\alpha,\rho)}{\alpha} + \frac{\theta[E(\alpha,\rho)]}{\alpha} \right), \tag{3.57}$$

where $\nu = m/k$, and $\theta[E(\alpha,\rho)] = \ln(e^{\rho[E(\alpha,\rho)-k\ln 2]} - 1)^{1/\rho}$.

This bound can be converted to a lower bound on the free distance of binary convolutional codes by considering the n-dimensional hypercubic constellation used to send binary code sequences. Example 3.3.1 gives an expression for the bound applied to rate 1/2 binary convolutional codes.

Example 3.3.1 *For a rate 1/2 binary convolutional code, we use a QPSK constellation. Thus, by considering a code where each symbol of the constellation can be sent with equal probability, we obtain*

$$E(\alpha,\rho) = -\ln(\frac{1}{4} + \frac{1}{2}e^{-2\alpha} + \frac{1}{4}e^{-4\alpha}). \tag{3.58}$$

As m becomes large, (3.57) becomes

$$d^2_{free} > \max_{\substack{E(\alpha,\rho) > \ln 2 \\ \alpha \geq 0}} \left(m\frac{E(\alpha,\rho)}{\alpha} \right), \tag{3.59}$$

so

$$\lim_{m\to\infty} \frac{d^2_{free}}{m} > 1.57, \tag{3.60}$$

which corresponds to

$$\lim_{m\to\infty} \frac{d_{free_H}}{m} > .785, \tag{3.61}$$

where d_{free_H} is the corresponding minimum free Hamming distance between binary symbols sent with the QPSK constellation. (3.61) is the same bound as the direct derivation for binary convolutional codes by Costello [39].

This bound is particularly interesting since it depends on the signal constellation used to construct the trellis code. Therefore, it could be interesting to study how to

construct the constellation in order to optimize the lower bound. In particular, the difference between the bound for the optimized constellation and the bound for the n-dimensional hypercube corresponding to the transmission of binary code sequences could show how much improvement is expected by constructing trellis codes over the real numbers as opposed to the binary numbers.

However, as mentioned in the beginning of this section, Chao and Chiu [41] have found a flaw in Rouanne and Costello's derivation of the bound. Specifically, the way

$$T_{\alpha,\rho}(c) = \sum_{v \in C} \left(\sum_{v'} e^{-\alpha d_E^2(v,v')} \right)^{\rho} \tag{3.62}$$

is defined in (3.52) is inappropriate, since it is possible to find an infinite number of pairs (v, v') with $d_E^2(v, v') \leq d$, by simply taking all possible semi-infinite paths starting after v and v' have remerged as shown in Figure 3.6. Thus, $T_{\alpha,\rho}(c)$ may not be finite, which then makes it difficult to upper bound in the rest of the derivation. Although Chao and Chiu presented a new correct version of the bound, it is much weaker than the original bound, and we will not derive it here.

Yet, Chao and Chiu found that the derivation does apply to linear convolutional codes, since the derivation can then be based on a single transmitted code sequence per code, which explains why this bound meets the binary convolutional code bound derived by Costello in Example 3.3.1. The original bound is also valid for geometrically uniform codes, which we will describe into more details in chapters 4 and 5.

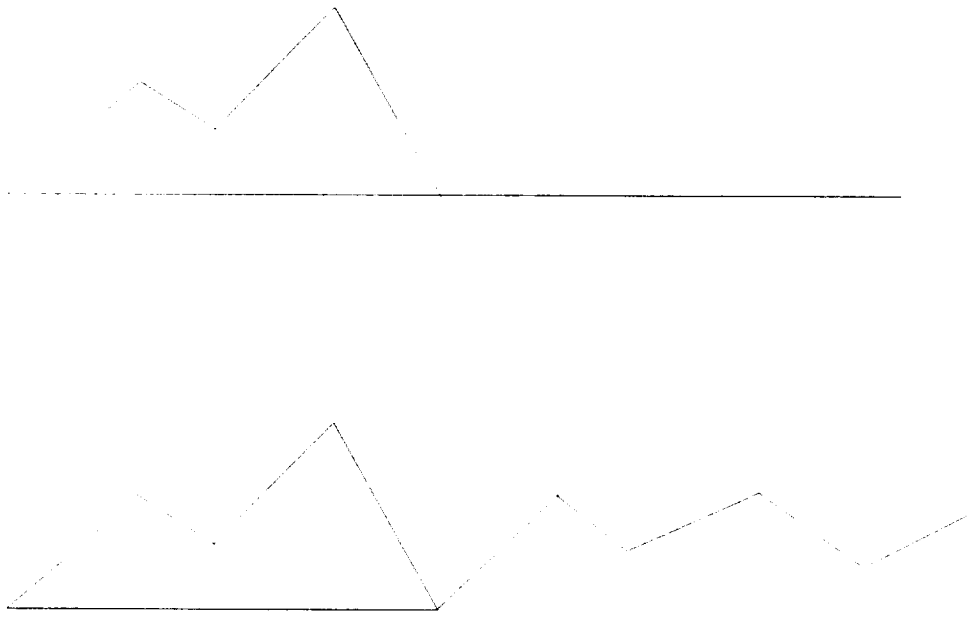

Figure 3.6: Different trellis paths separated by the same distance.

3.4 Conclusion

In this chapter, we have studied asymptotic and non-asymptotic bounds on the minimum distance of binary and real number block and trellis codes. It was observed that asymptotic lower and upper bounds are greater for real number codes than binary codes, especially as the rate of the code increases. For the non-asymptotic case, the difference becomes smaller as the constraint length of the trellis code increases. The sphere packing problem was introduced to derive the bounds but also to help us understand the main idea of coding. It involves positioning codewords in the Euclidean space of finite (block codes) or infinite (trellis codes) dimensionality. While binary codes are constructed on a specified set of positions in the space, determined by the binary Z^n lattice, real number codes can be constructed by using the entire space, thus providing a better way of achieving the best sphere packing in that space. The

following chapters will consider the construction of trellis codes in the real field, and will introduce a geometric approach to help constructing and understanding how to achieve a good sphere packing with trellis codes.

CHAPTER 4

LATTICES AND TRELLIS CODES CONSTRUCTIONS

Real number trellis codes were first introduced by Ungerboeck [3] as a combined coding and modulation scheme improving error performance without sacrificing data rate or requiring more bandwidth. In this chapter, the concept of this coding scheme will be introduced. In particular, concepts as set partitioning of a signal constellation, the use of lattices to design the signal constellation, and multi-dimensional codes will be described. The description of this subclass of real number codes will lead to the general class of real number trellis codes.

4.1 Channel coding with expanded signal constellations

Usual coding schemes consider the design of binary block and convolutional codes independently from the modulation used to transmit them, since the squared Euclidean distance between codewords transmitted using BPSK or QPSK constellations is proportional to the Hamming distance between codewords in the binary field (see Section 1.4). This is due to the fact that the binary numbers 0 and 1 can directly be

mapped onto the two points of the BPSK, or for the two-dimensional case, 00, 01, 10, and 11 can be mapped onto the four points of the QPSK such that the Hamming distance between the binary numbers and the squared Euclidean distance between their representative modulated points is proportional.

Ungerboeck's idea is to use N-dimensional signal constellations with $\mathcal{N} > 2^N$ points, such that more binary digits can be transmitted per time unit. However, without coding, such a scheme deteriorates the transmission, since the Euclidean distance between signal points becomes smaller. By adding a block code or a convolutional code in front, it is possible to obtain an overall Euclidean distance between codewords or code sequences greater than the distance between points in the original signal constellation with $\mathcal{N}$ points. The general block diagram for trellis codes on expanded signal constellation is shown in Figure 4.2. The k binary information bits are divided into $\tilde{k}$ bits entering the rate $\tilde{k}/(\tilde{k}+1)$ binary convolutional encoder, and $p = k - \tilde{k}$ bits entering directly the mapper. The $\tilde{k}+1$ encoded bits are then joined to the other p uncoded bits and mapped to one of the $2^n = 2^{k+1}$ points a_i of the N-dimensional signal constellation. The trellis structure is then constituted of diverging paths corresponding to the encoded bits and parallel branches corresponding to the uncoded bits as shown in Figure 4.1. Note that the binary convolutional encoder also adds redundancy in terms of number of bits at the output. However, by expanding the signal constellation from $\mathcal{N} = 2^k$ points to $2\mathcal{N} = 2^{k+1} = 2^n$ points, we can map this extra bit back to the N dimensional constellation.

Note that the points in the constellation can be represented by their N real number coordinates. Therefore, the binary code sequences mapped on the constellation

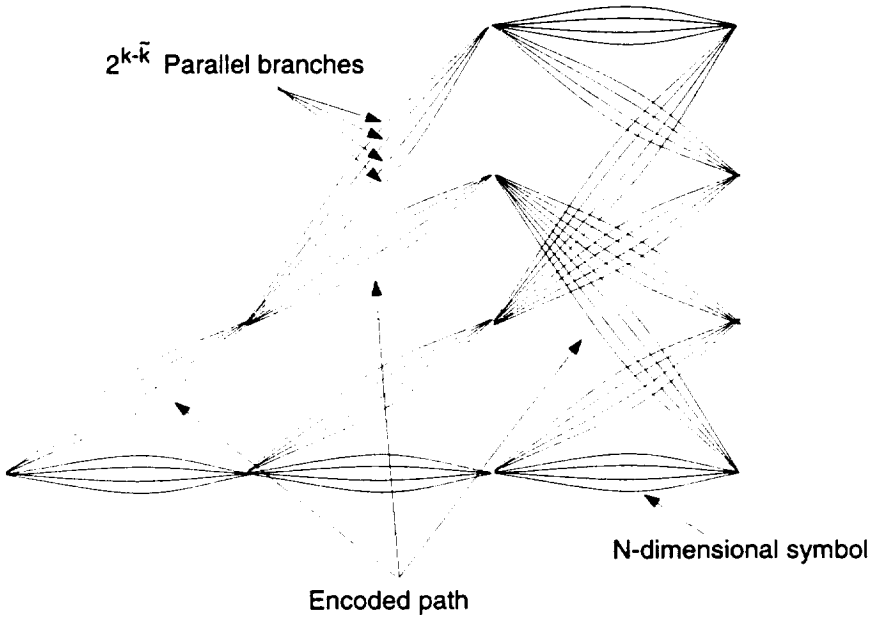

Figure 4.1: Trellis structure of a trellis code on an expanded signal constellation.

can also be directly viewed as real number code sequences, and the distance one is interested in is now the squared Euclidean distance between the code sequences instead of the Hamming distance between their binary representatives. Therefore, the redundancy added by the code on the binary field does not affect the overall rate R of the code defined by

$$R = \frac{k}{N}. \tag{4.1}$$

The purpose of constructing the real number code sequences by decomposing the encoder into a binary encoder and a mapper is to use algebraic properties on the binary field to construct good binary codes, and then use a technique introduced by Ungerboeck called set partitioning to design good mappers. Since a good binary code is a code with a large minimum Hamming distance, a good mapper is designed to yield a large minimum squared Euclidean distance between the signal points associated to

large Hamming distance between their binary representation. While this mapping is simple to realize when the number 2^n of points in the constellation is less than or equal to 2^N (proportionality between Hamming distance and squared Euclidean distance), it is not as simple for constellations with more than 2^N points. The set partitioning technique described in section 4.2 provides a technique to fix the relationship between the Hamming distance between binary code sequences and the squared Euclidean distance between the corresponding signal points.

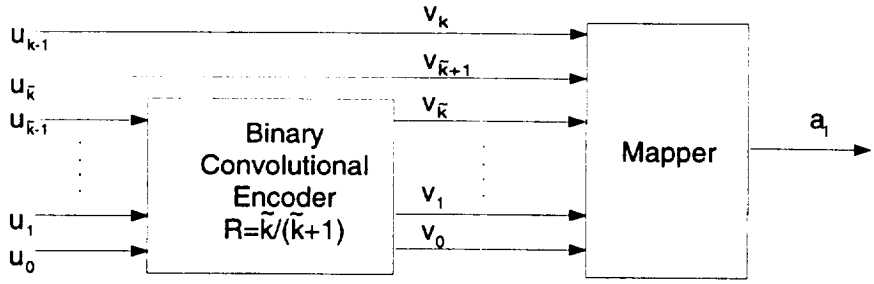

Figure 4.2: Expanded signal constellation trellis encoder block diagram.

Although both binary block codes and convolutional codes can be used in conjunction with an expanded signal constellation, it is very advantageous to be able to use a soft decision decoder in order to use the entire coding gain obtained from the minimum squared Euclidean distance of the code. Whereas algebraic techniques can efficiently be used to decode block codes over their construction field, trellis codes can be decoded with soft decision by using the Viterbi algorithm. This present advantage of trellis codes over block codes motivates us to focus on trellis codes in the remainder of the dissertation, when constructing real number codes.

The coding gain introduced in Section 1.4.2 corresponds to the savings of energy when using a coded system as opposed to using an equivalent uncoded system. Specif-

ically, for trellis coding with expanded signal constellation, the coding gain can be computed as

$$G_{C_n/C_{n-1}} = 10 \cdot \log_{10} \frac{d^2_{free}(C_n)}{\Delta^2_0(C_{n-1})}, \tag{4.2}$$

where $d^2_{free}(C_n)$ is the minimum free squared Euclidean distance of the trellis code used on the expanded constellation with 2^n points and $\Delta^2_0(C_{n-1})$ is the minimum squared Euclidean distance between any of the 2^k points of the constellation before expansion.

Example 4.1.1 *Let us expand a QPSK ($k = 2$) constellation into an 8PSK ($n = k + 1 = 3$) constellation. For the QPSK constellation,*

$$\Delta^2_0(C_2) = 2. \tag{4.3}$$

Since two information bits can be transmitted during a time interval T, we use a 4-state, rate $1/2$ convolutional code combined with an uncoded bit as shown in Figure 4.3, thus yielding 3 bits into the mapper. The corresponding trellis is shown in Figure 4.4. These three bits are then mapped into the 8PSK constellation shown in Figure 4.5. The squared free Euclidean distance between any two code sequences in the trellis is 4, thus yielding a coding gain

$$G_{8PSK/QPSK} = 10 \log_{10}(4/2) = 3 \ dB. \tag{4.4}$$

Note that the coded and uncoded systems are equivalent since they both transmit 2 bits per time interval.

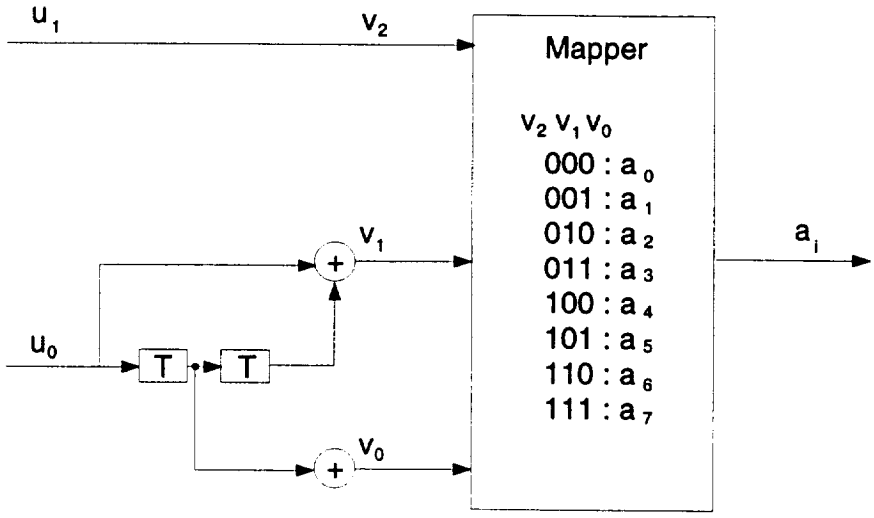

Figure 4.3: 4 state, rate 2/3 8PSK trellis encoder.

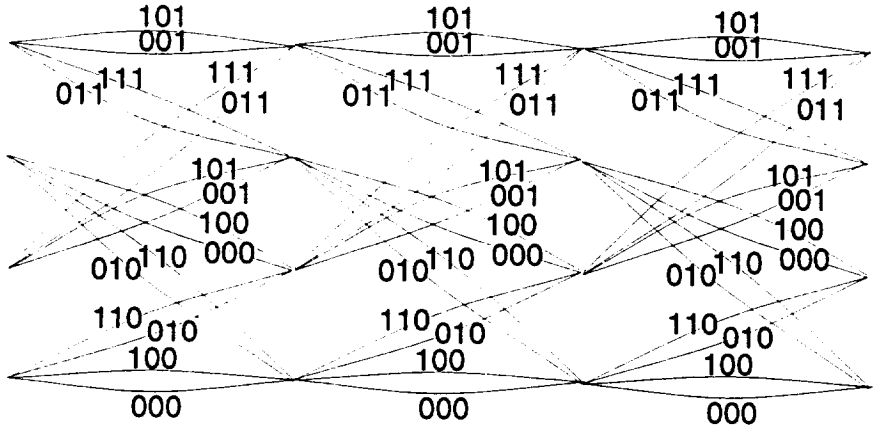

Figure 4.4: 4 state, 1 uncoded bit, rate 2/3, trellis diagram.

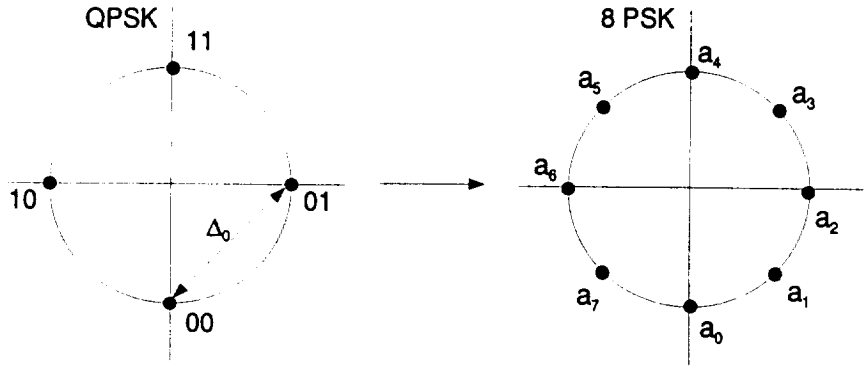

Figure 4.5: QPSK and expanded 8PSK constellations.

4.2 Set partitioning techniques and code constructions

It was mentioned in the previous section that the mapper realizes a one-to-one mapping from binary n-tuples into a signal point of the constellation and tries to impose a fixed relationship between the Hamming distance between binary n-tuples and the squared Euclidean distance between their representative signal points. In this section, a method first introduced by Ungerboeck [3] called set partitioning is described.

4.2.1 Set partitioning

A given signal constellation is successively decomposed into subsets of points so that the minimum squared Euclidean distance between any two points in a subset increases along the decomposition. The binary labeling associated to each point then depends on which subset the point belongs to.

Let Δ_i denote the minimum distance between any two points of a subset after i decompositions. Note that Δ_0 denotes the minimum Euclidean distance between any two points of the entire signal constellation, Figure 4.6 shows how the mapping given in Figure 4.3 is derived by set-partitioning the 8PSK constellation. The parallel branches of the trellis are then assigned with points from subsets with the largest possible minimum distance (Δ_2 in this case), and branches joining in and originating from the same state are assigned with points from subsets with smaller distance (Δ_1 in this case).

The squared Euclidean distance between the signal points sent on two parallel

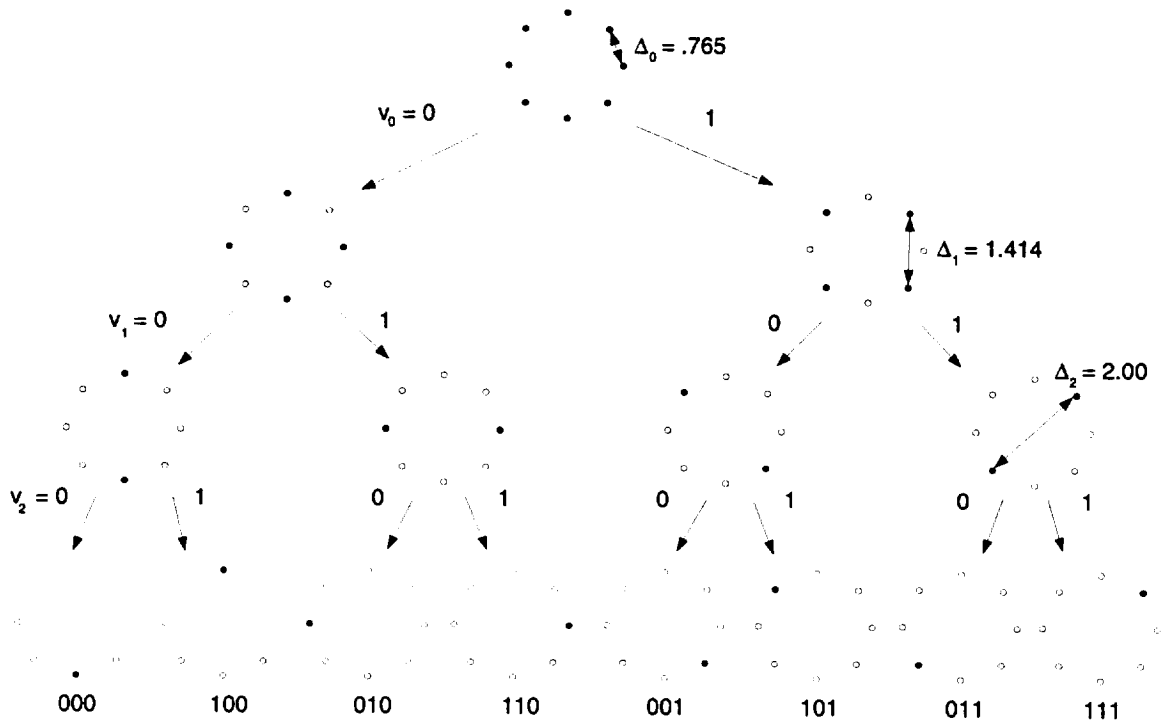

Figure 4.6: Set partitioning of 8PSK constellation.

branches (see Figure 4.4) is $\Delta_2^2 = 4.0$, and the minimum free squared Euclidean distance between any two diverging and remerging paths in the trellis is $2\Delta_1^2 + \Delta_0^2 = 4.585$. Therefore, the minimum free squared Euclidean distance of the code is 4.0. More generally, the minimum free squared Euclidean distance of a trellis code is given by

$$d_{free}^2 = \min[\Delta_{\tilde{k}+1}^2, d_{free}^2(\tilde{k})], \tag{4.5}$$

where $\Delta_{\tilde{k}+1}$ is the minimum Euclidean distance between parallel branches and $d_{free}(\tilde{k})$ denotes the minimum distance between diverging and remerging paths in the trellis.

4.2.2 Convolutional code construction

The trellis code construction is simple for small trellis topologies and small signal subsets, but it becomes more tedious as the constraint length increases and a systematic code construction is necessary. Once the signal constellation is set partitioned, it is indeed necessary to find the best binary convolutional code associated to a given mapper. Since the convolutional code is of rate $\tilde{k}/(\tilde{k}+1)$, it is possible to realize it in feedback form instead of feedforward as defined in Section 2.1. The feedback form of an encoder is directly related to the parity-check matrix $H(D)$ of the convolutional code defined by

$$G(D)H^T(D) = 0, \tag{4.6}$$

where $G(D)$ is the generator matrix of a code as defined in Section 2.1. The purpose of using $H(D)$ instead of $G(D)$ in our case is that $H(D)$ is defined by only $\tilde{k}+1$ polynomials whereas $G(D)$ must be defined by $\tilde{k}\times(\tilde{k}+1)$ polynomials. This decreases considerably the time to search for the optimal binary convolutional code.

Example 4.2.1 *In example 4.1.1, the generator matrix of the rate 1/2 convolutional code was*

$$G(D) = [1 + D^2, D], \tag{4.7}$$

so the parity-check matrix of the same code can be computed as

$$H(D) = [\frac{D}{1+D^2}, 1]. \tag{4.8}$$

The feedback encoder associated with $H(D)$ is shown in Figure 4.7.

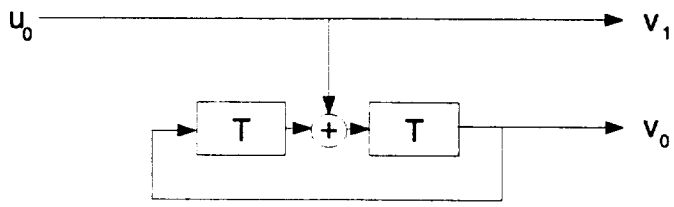

Figure 4.7: Feedback encoder for a 4 state, rate 1/2 binary convolutional code used with a trellis code.

More generally, from the parity-check matrix $H(D)$, code sequences $v(D)$ can be generated by

$$[v^k(D)\dots v^0(D)] = [u^{k-1}(D)\dots u^0(D)] \cdot \left[\begin{array}{c|c} & 0 \\ & \vdots \\ I_k & H^{\tilde{k}}(D)/H^0(D) \\ & \vdots \\ & H^1(D)/H^0(D) \end{array}\right], \qquad (4.9)$$

where I_k is the identity matrix of size $k \times k$. Note that the maximum number of binary coefficients to assign over any polynomial $H^i(D)$ for $0 \leq i \leq \tilde{k}$ is $\lceil \frac{m}{\tilde{k}} \rceil$, thus yielding an exhaustive search through $2^{\frac{m(\tilde{k}+1)}{k}}$ different matrices. For each matrix, the minimum free squared Euclidean distance of the code must be computed. However, while for linear codes, the free distance can be computed as the minimum weight of any non zero code sequence (see Section 1.4.3), the free distance of a trellis code with real number code sequences must be computed between any two possible sequences, which increases dramatically the computing time. This problem can be solved by making use of the subset distances resulting from the set partitioning of the signal constellation. The *Euclidean weight* $w(e)$ of a binary k-tuple error vector e can be

defined as

$$w(e) = \min_{a} d_E[a(u), a(u+e)], \tag{4.10}$$

where u can be any binary k-tuple input vector, $a(u)$ denotes its corresponding signal point, and d_E the Euclidean distance between the signal points. Therefore, the distance between any two sequences $u(D)$ and $u'(D)$ can be lower bounded by

$$d_E^2[a(u(D)), a(u'(D))] \geq w^2[u(D) + u'(D)], \tag{4.11}$$

where $w^2(e(D))$ is the sum of the squared weights of the components of $e(D)$ over time. Due to the set partitioning technique, the Euclidean weight $w(e)$ of a binary k-tuple e can be upper bounded by the subset distance $\Delta_{q(e)}$ corresponding to the number of trailing zeros $q(e)$ in e. For example, $q(e) = 2$ for $e = (e^{k-1}, \ldots, e^3, 1, 0, 0)$.

The free distance of the code can therefore be lower bounded by using Bahl and Larsen's algorithm [43], which computes the minimum free Hamming distance of binary linear convolutional codes, *i.e.* involving only the weight computation of non-zero sequences. That is,

$$d_{free}^2 \geq \Delta_{free}^2 = \min_{e(D) \neq 0} \Delta^2[e(D)], \tag{4.12}$$

where $\Delta^2[e(D)] = \sum \Delta_{q(e_i)}^2$, where e_i are the k-tuples composing $e(D)$. This lower bound is usually achieved and Δ_{free}^2 equals the free distance due to the symmetries in the structure of the constellations. In other words, due to the fixed relationship between the Euclidean distance between two points in the constellation and the Hamming distance between their corresponding binary labels, it is possible to compute the free distance by examining all non-zero sequences and associate a Euclidean weight

to each possible label, instead of a binary Hamming weight as it would be done with a convolutional code. Note, however, that the Euclidean weight of a symbol, defined in (4.10), is determined by a minimum over all symbols separated by the same binary label. This implies that some sequences have the same sum in the binary field, but not the same Euclidean distance. For most of the codes constructed by Ungerboeck on the usual symmetric QAM modulation schemes, the minimum distance is the same from all possible sequences. This observation has lead to definition of geometric uniformity [15] for real number codes as an extension of the linearity property for binary convolutional codes.

Definition 4.2.1 *A code with the same distance spectrum from any code sequence is called* geometrically uniform.

In other words, for a geometrically uniform code, the set of distances between any given code sequence and all the others is the same. In particular, the set of distances from the all zero sequence is the same than from any other sequence, thus allowing one to compute the free distance by taking the all zero sequence as reference, as in the Bahl and Larsen's algorithm to compute free distance. All binary linear convolutional codes are geometrically uniform since there is a proportional relationship between the Hamming distance in the binary field and the Euclidean distance when using BPSK or QPSK constellations. In chapter 5, we will study a new approach for constructing geometrically uniform codes from a geometrical point of view, rather than algebraic. We will see that the best geometrically uniform codes are not necessarily constructed on the usual signal constellations. In fact, while Ungerboeck's technique is very

efficient to construct good codes, it's goal is not to construct optimal codes, but rather to improve the transmission over bandwidth limited channels in a simple way.

Different questions remain about the optimality of this construction technique. First, are the expanded constellations optimal for use with trellis codes ? It will be seen in the following sections and chapter that the constellation design can be improved. Second, is the binary set partitioning technique the best for designing a mapper used with a convolutional code ? The next section will show that in some cases, binary set-partitioning is not always the best, and that in fact, it should depend on the topology of the trellis and the dimensionality of the signal constellation. Finally, are binary linear convolutional codes optimal for reaching the best minimum Euclidean free distance ? We will see in the following sections that for some constellations, non binary convolutional codes can bring some advantages over binary codes for distance and performance analysis purposes.

4.3 Lattices and non-binary set partitioning

It was first observed by Calderbank and Sloane [44] that most QAM signal constellations can be seen as finite sets of points taken from an infinite lattice. Pursuing this approach, Forney [45, 46] defined the general class of *coset codes* based on the set partitioning of lattices into sublattices for mapping onto a convolutional code.

Let Λ be an N-dimensional lattice, and Λ' a sublattice of Λ, that is a subset of points of Λ which is itself a lattice. This creates a *partition* Λ/Λ' of Λ into *cosets*. These cosets are translated version of Λ' within Λ, *i.e.* one point of the coset defines

the entire subset of points. It is then simple to have a convolutional encoder to select one of the cosets of the main lattice. An often used lattice is the Z^N lattice, or *binary lattice*, defined by the Cartesian product of the one-dimensional set of integer Z taken from the real Euclidean one-dimensional line.

Example 4.3.1 *Consider the Z^2 lattice and decompose it into 4 cosets, which are themselves $2Z^2$ lattices as shown in Figure 4.8. This partitioning technique replaces the binary partitioning tree introduced by Ungerboeck. Note that in this case, a binary tree partitioning leads to an equivalent partitioning in two successive operations.*

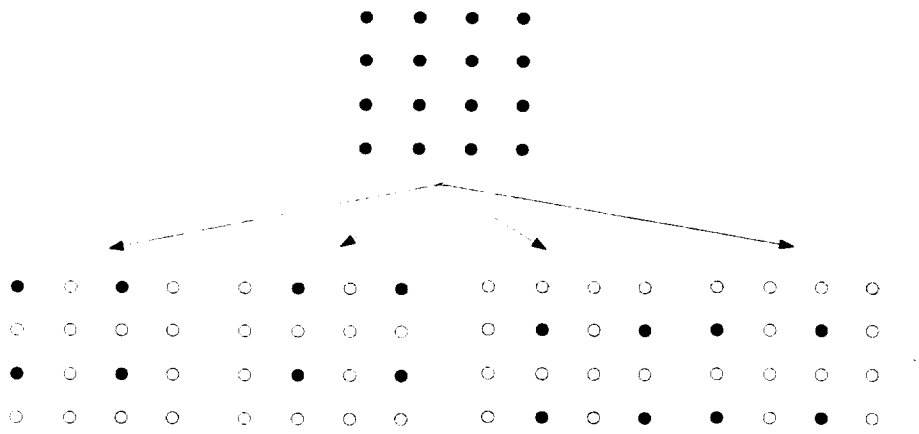

Figure 4.8: $2Z^2$ cosets resulting from the $Z^2/2Z^2$ partition of the Z^2 lattice.

However, the purpose of using lattices to construct trellis codes is to be able to use lattices with better sphere packings than the usual Z^2 lattice, and to partition by using known decomposition of lattices into composing sublattices. Lattices up to 24 dimensions are well known and their decomposition into sublattices can easily be found in the literature [47]. Most of these lattices are binary, *i.e.* can be decomposed into Z^l lattices with $l \leq N$. Therefore, signal points can be put on a trellis topology

which usually presents a number of diverging branches from each state equal to a power of 2. However, we will see in Chapter 5 that there exists geometrically uniform structures that are not lattices, even though they are composed of lattices. These structures can lead to better codes than the trellis codes constructed on known lattices. In other words, it is not always the best packings in N dimensions which lead to the best trellis code infinite dimensional sequences. Another purpose of the non binary tree set partitioning is that it is not always optimal, as example 4.3.2 shows.

Example 4.3.2 *Consider a binary Z^3 lattice within cubic boundaries as shown in Figure 4.9. Suppose we want to decompose the set into 4 cosets of two points. The binary set partitioning does not lead to the best cosets in terms of minimum distance, whereas a direct four-way partitioning leads to the 4 diagonals of the cube, which is the optimal partitioning.*

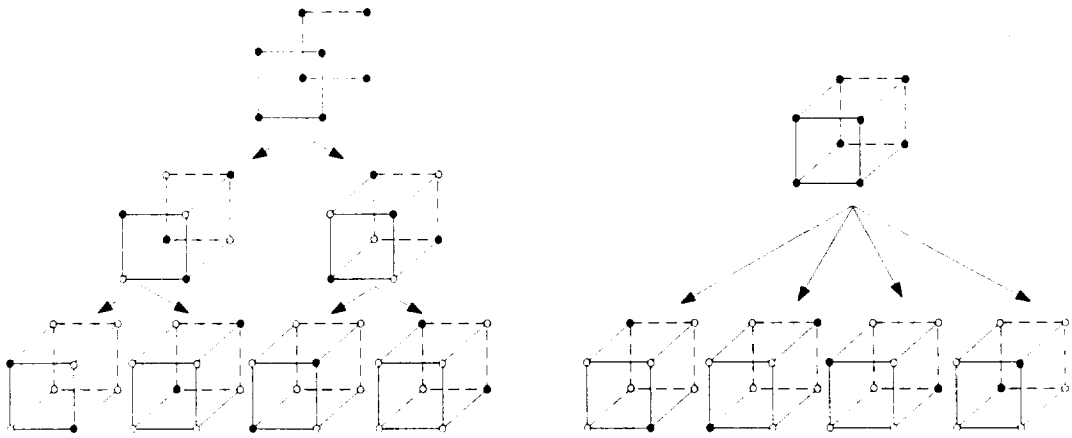

Figure 4.9: Binary and four-way partitioning of a 3D cube.

Example 4.3.2 presented a trellis code on a N-dimensional constellation with $N > 2$. These trellis codes are called *multi-dimensional* trellis codes, since they

cannot be sent in one time unit T on a modulation scheme using only one frequency. Usually, such codes are constructed on constellations with N even, which allows one to decompose the transmission into a finite number of time units T. One way of constructing constellations with N even is to take the Cartesian product of 2-dimensional constellations.

4.4 Trellis codes on multi-dimensional constellations

Wei [48] was the first to investigate the construction of trellis codes over multi-dimensional constellations. Since block codes present a better minimum distance as n and k increase, the rate k/n being constant, it seems interesting to increase k and N for trellis codes as well. In particular, the parallel branches of the trellis as well as remerging trellis paths correspond to rate k/N block codes. So the minimum distance between parallel branches should increase with N. However, (4.5) specified that the overall minimum free Euclidean distance of the code is given by the minimum between the path distance and the parallel distance. Thus, increasing the parallel distance does not necessarily increase the overall minimum free distance of the code, unless it is possible to decrease $\tilde{k}$ while maintaining k. Smaller $\tilde{k}$ leads indeed to less connectivity in the trellis, *i.e.* longer paths, but larger p, *i.e* more parallel branches.

Another purpose for constructing multi-dimensional trellis codes is to obtain a larger range of transmission rates. When using 2D constellations, it is indeed possible to transmit an integer number of information bits per time unit T. 4D constellations allow one to also transmit k information bits during 2 time units T, thus yielding

transmission rates of $k/2$ bits per time unit.

The partitioning of multi-dimensional constellations obtained from Cartesian products of lower-dimensional constellations can be done by partitioning the constituent 2D lattices into subsets $A, B, C, \ldots$. Different 4D subsets can then be constructed from the 2D subsets by making pairs of 2D subsets such as $(A, A), (A, B), (A, C), \ldots$, $(B, C), \ldots$. Those 4D subsets present the same minimum distance Δ_i^2 than their constituent 2D sets. This partitioning technique provides a simple way of partitioning constellations of higher dimensions. However, since these constellations are obtained from the Cartesian product of lower dimensional constellations, they do not necessarily correspond to the best sphere packing in higher dimension. This was studied by Wei [48] for the 8D case. Two constellations constructed from the rectangular Z^8 lattice and the best lattice packing in 8 dimensions E_8 yielded two codes with the same coding gain over uncoded. We will see in chapter 5 that such an observation is related to how constellations must be constructed for optimizing the free distance of trellis codes. They should indeed not necessarily be taken from lattices, but rather from superimposed lattices.

Also note that the binary set-partitioning is particularly adapted to lattices of dimensionality equal to a power of 2. Example 4.3.2 showed a 3D constellation for which the binary set partitioning is not necessary optimal. Similarly, for other constellations with dimensionality different from a power of 2, the same observation can be made. In particular, Pietrobon [49] noted that two different set partitioning can be made on a 6D constellation, and depending on the trellis topology, one of the two is preferable for obtaining the best minimum free distance.

4.5 Non binary convolutional codes

In the previous sections, we have shown that Ungerboeck's technique for constructing trellis codes over the real field provides good but not necessary optimal codes. In particular, it was noted that the best constellation used with block codes is not necessary optimal for trellis codes. Also, the binary set partitioning technique introduced by Ungerboeck is not always optimal, depending on the dimensionality of the constellation and the topology of the trellis code. In this section, we study the optimality of the field for constructing the underlying binary convolutional code used before the mapper.

This question was first studied by Massey and Mittelholzer [14] for convolutional codes mapped on PSK constellations. It is indeed shown that trellis codes over M-PSK constellations constructed with convolutional codes over Z^M exhibit the same properties than binary convolutional codes, in particular, the distance between two code sequences can be computed as the distance between the all zero sequence and another. Thus, the resulting trellis code is linear with respect to the Euclidean distance. Specifically, a signal point can be represented by the complex number W_M^i for $0 \leq i \leq M-1$, where $W_M = e^{j2\pi/M}$. The squared Euclidean distance between two signal points i and j is thus

$$d_E^2 = |W_M^j - W_M^i|^2 = |1 - W_M^{j-i}|^2. \tag{4.13}$$

Therefore, defining the *phase weight* of the signal point i by

$$w(i) = |1 - W_M^i|^2, \tag{4.14}$$

the *phase distance* between i and j can be computed by

$$d_E^2(i,j) = w(j-i). \tag{4.15}$$

Thus, considering two sequences v_1 and v_2 with symbols drawn from an M-PSK constellation, the Euclidean distance between v_1 and v_2 can be computed as the weight $w(v_1 - v_2)$, where the weight of a sequence is computed by summing the weights of each component in the M-PSK constellation.

In particular, this provides a geometrically uniform code, that is, the minimum free squared Euclidean distance of the code can be computed from the all zero sequence, just as for binary linear convolutional codes. Loeliger [16] extended this study by describing all constellations for which convolutional codes can be constructed on groups that match the constellation in terms of linearity properties. The main contribution of this paper is that a signal set is matched to a group, if and only if it is a translate of what Slepian [50] calls a "group code for the Gaussian channel", that is a signal set obtained from orthogonal transformations applied to a point of $\mathbb{R}^N$. More recently, Trott [51] studied this concept with isometry codes, *i.e*, a group of sequences of isometries of the Euclidean space $\mathbb{R}^N$, and applied it to trellis codes.

In general, these new approaches of coding in the real field are trying to define an algebraic structure matched to the geometrical problem of sphere packing, in order to find techniques to design and analyse codes. In the next section, we are defining the most general class of real number trellis codes, and consider a direct approach for optimizing the set of code sequences without considering any algebraic underlying code. We will construct some codes with better minimum free squared Euclidean

distance than previously constructed trellis codes. However, this technique will be limited by the number of parameters that need to be optimized, and chapter 5 will concentrate on constructing real number trellis codes with the constraint of being geometrically uniform. This new technique will actually lead to the same codes than those constructed in the following section, and provide an efficient way of reducing the search, thus allowing us to find longer and more complex codes.

4.6 General class of real number trellis codes

Although the algebraic structure of the binary field provides efficient techniques for constructing block codes and decoding received codewords, it requires a bandwidth increase or a data rate decrease, due to the number of redundant bits to add to the information for protection against noise. This redundancy is not necessary when using the real field instead, and code rates can be lower or greater than 1, while still protecting the information against noise. This brought researchers such as Ungerboeck [3] to find efficient techniques for constructing good trellis codes over the real field. As it was seen in previous sections, his technique and improvements by other researchers have lead to very good codes, but the decomposition of the code into an algebraic binary convolutional code and a mapper into the real field makes it impossible to prove the optimality of the codes found. The purpose of this section and the following chapter is to start from the most general type of trellis codes over the real field, and optimize the construction of code sequences over this structure. We will then compare the new codes obtained and compare them with the codes constructed

with the usual technique.

4.6.1 Definition

A real number trellis code is a set of code sequences over the real field related by a trellis structure. Let C be a rate k/n constraint length m real trellis code with 2^m states and p parallel branches. C is defined by a set of real numbers a_i for $i = 0 \ldots n2^{k+m} - 1$. The binary information sequence $U(D)$ defines a path through the trellis and the real code sequence $V(D)$ is the set of real labels on that path. Starting at a given state s ($s = 0, \ldots, 2^m - 1$) in the trellis, an information symbol j ($j = 0, \ldots, 2^k - 1$) produces n outputs $a_{sn2^k+jn}, \ldots, a_{sn2^k+jn+n-1}$, as shown in Figure 4.10. A general encoder for a real trellis code is shown in Figure 4.11. The set of shift-registers creates a finite-state machine, which expanded in time corresponds to a trellis structure. The memory element then selects the label corresponding to the given state of the trellis and the k binary inputs at this time.

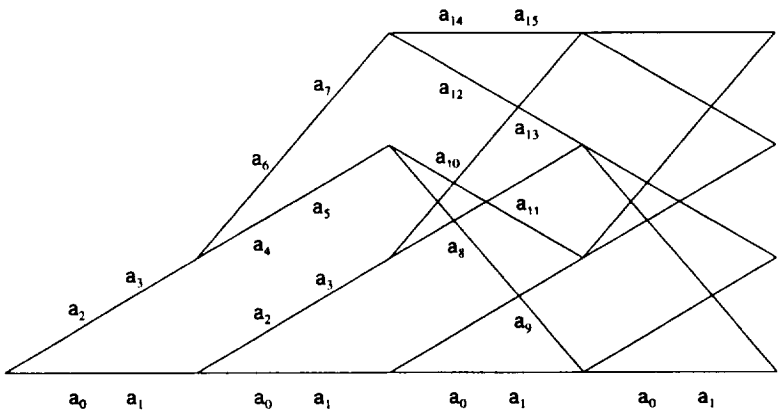

Figure 4.10: Trellis for a rate 1/2 constraint length 2 real trellis code with no parallel branches.

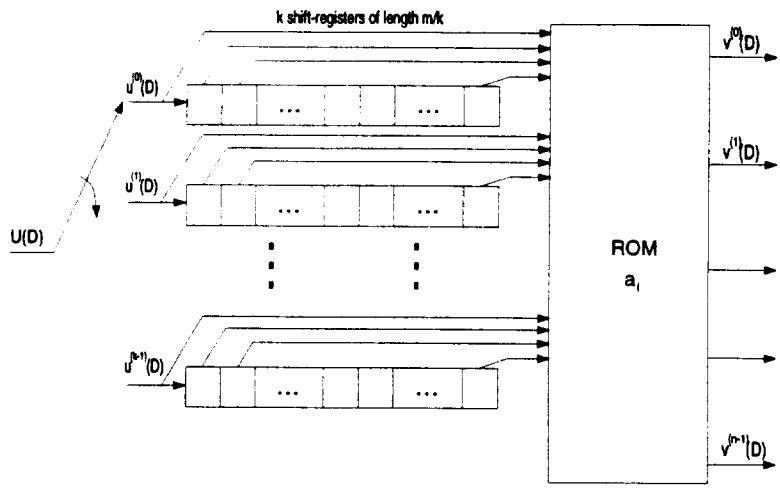

Figure 4.11: Real trellis encoder.

Note that a code is a set of code sequences in $\mathbb{R}^n$. The encoder or the labeled trellis then maps each code sequence to a different information sequence. Even though we consider binary information sequences, any other type of indexing for the information sequence is possible. This explains the name of *real number trellis codes.*

The modulation scheme associated with the transmission of a real number trellis code is the n-dimensional constellation with signal point coordinates given by the n-dimensional branch outputs a_i. This is comparable to sending binary digits on a BPSK constellation ($n = 1$), or pairs of binary digits on a QPSK constellation ($n = 2$). Therefore, we can keep the usual definition of the rate R of a trellis code, that is

$$R = \frac{k}{n}. \tag{4.16}$$

Since we now consider $n = N$, in the remainder of this thesis, we will use n instead of N to denote the dimensionality of the constellation. Note that Ungerboeck specifies the rate of a trellis code by indicating the rate of the underlying convolutional code

and what type of constellation is used by the mapper. In our case, we do not specify any a priori constellation, but rather a number of dimensions for the Euclidean space on which code sequences are constructed. Ungerboeck also assumes that 2 dimensions can be transmitted during one time unit by using a QAM constellation. Therefore, the transmission rate for our trellis codes is $2R$ bits per time unit T.

For the purpose of comparing coding schemes with each other, it is necessary to normalize the average energy used to transmit code sequences. This is done by normalizing the coefficients a_i for $i = 0 \ldots n2^{k+m} - 1$ by

$$\frac{1}{2^{k+m}} \sum_{i=0}^{n2^{k+m}-1} a_i^2 = 1, \tag{4.17}$$

which corresponds to sending code sequences with an average power of 1. We also abandon the concept of comparing a given constellation and a code expanded constellation, but rather compare the optimal coded scheme in a certain dimension with the uncoded scheme in the same number of dimensions, as it is done for binary codes. Although we should compare optimal coded schemes with optimal uncoded schemes, *i.e.* the best packing of 2^k points in n dimensions, we use the Z^n lattice, and take the 2^k points with smallest energy as uncoded scheme. As k increases, this may not give the best packing and the coding gain shown may be slightly larger than the coding gain obtained from the best uncoded scheme. Also, for $k < n$, the best packing for the uncoded scheme is obtained by using the best rate k/n block code, which is not always obvious to find. Therefore, when using the Z^n lattice, we can simply compute the minimum distance between uncoded 2^k points by computing the average power of all 2^k points with smaller energy separated by distance 1 in the Z^n lattice. Example

4.6.1 describes the computation for a two dimensional constellation.

Example 4.6.1 *Figure 4.12 shows a constellation with* $2^7 = 128$ *points in an* $n = 2$*-dimensional space. For this set of points, the average energy is 20.5, so the minimum squared Euclidean distance between points with average energy 1 is* $d_{min} = 1/20.5 = .049$*. Suppose the free squared Euclidean distance of a trellis code of rate* $7/2$ *is* d^2_{free}*, then the coding gain by using this code over an uncoded scheme is given by*

$$G_{dB} = 10 \log_{10} \frac{d^2_{free}}{.049} \tag{4.18}$$

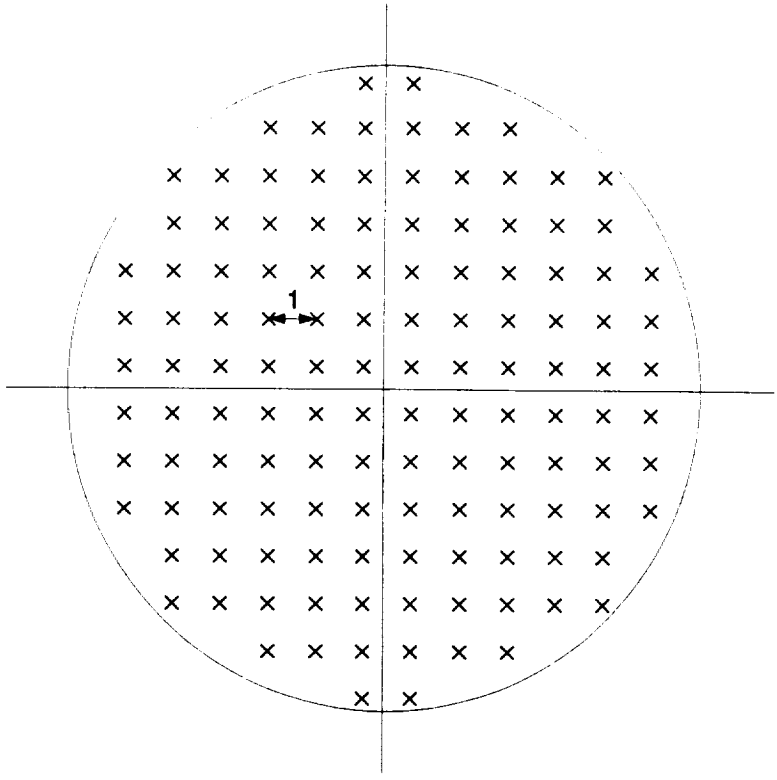

Figure 4.12: 128 points taken from the 2-dimensional Z^2 squared lattice.

Table 4.1 gives a list of the minimum squared Euclidean distances obtained by using this constellation type for different dimensionalities n and number of points k,

k	n	d^2_{min}
1	2	4.00
2	2	2.00
3	2	0.80
4	2	0.40
5	2	0.19
6	2	.095
7	2	.049
1	3	4.00
2	3	2.67
3	3	1.33
4	3	0.57
5	3	0.44
6	3	0.26
7	3	0.17
1	4	4.00
2	4	2.00
3	4	2.00
4	4	1.00
5	4	0.50
6	4	0.40
7	4	0.29

Table 4.1: Table of minimum distance for Z^n lattice constellations

when $k > n$. When k is lower than n, we use the best rate k/n binary block code, which corresponds to points taken in the Z^n lattice also. For $n = 2$ and $k = 3$, 4, 5, we use the standard 8AMPM, 16QAM, and 32AMPM for the uncoded constellation. Note that the distance reached in this table by 2^k points in n dimensions can be higher than the upper bound on the distance reached in Table 3.2 in Chapter 3, since we must recall that the upper bound was given as the maximum distance divided by the maximum energy of the constellation, whereas here, we are interested in the maximum distance divided by the average energy of the constellation.

4.6.2 Construction of real number trellis codes

While algebraic techniques proved to be useful for constructing binary block codes, we saw in Chapter 2 that binary convolutional codes need to be constructed by exhaustive search, since no algebraic method so far has lead to good codes [19, 20, 21, 22]. The exhaustive search consists in examining all possible binary code generators and selecting the code that yields the largest minimum free distance. This method is also employed to construct real number trellis codes, when using Ungerboeck's decomposition and selecting the best underlying binary convolutional code. However, when constructing directly real number trellis codes, it is necessary to find the $n2^{k+m}$ real numbers which optimize the free distance of the code. This presents numerous difficulties. First, the number of parameters to choose is much larger than for a binary linear convolutional code, for which only the generator needs to be specified ($m+1$ coefficients). Second, the density of the real field does not allow us to search exhaustively for the best real trellis codes, as is done for binary convolutional codes. And third, the free distance needs to be computed from all code sequences, since the code is not necessary linear.

Formally, if y_τ is a code sequence of C of length τ, and $e_{y_\tau}(C)$ is the set of all possible sequences of length τ different from y_τ, then the free distance of the code C is defined as

$$d_{free}(C) = \min_{\substack{\lceil \frac{m}{k-p} \rceil + 1 \leq \tau \\ y_\tau \in C, y' \in e_{y_\tau}(C)}} d_E(y, y'). \tag{4.19}$$

Since it is necessary to compute the minimum free squared Euclidean distance from all code sequences, we cannot use the Bahl and Larsen's algorithm [43] to compute

the free distance. To compute the distance, we use the row distance d_l parameter which was described in Chapter 2 as

$$d_l = \min_{\substack{U(D)\neq 0 \\ \deg U(D)\leq l}} w(V(D)), \tag{4.20}$$

where $U(D)$ was the binary information sequence specifying the path through the trellis, and $w(V(D))$ was the weight of the binary code sequence. In the real field, we extend this definition to the Euclidean row distance $d_{U_0}(l)$ of order l from a given information sequence $U_0(D)$, that is,

$$d_{U_0}(l) = \min_{\substack{U(D)\neq U_0(D) \\ \deg U(D)\leq l}} d_E(V(D), V_{U_0}(D)), \tag{4.21}$$

where $d_E(V(D), V_{U_0}(D))$ denote the Euclidean distance between the real number code sequence $V(D)$ and the code sequence $V_{U_0}(D)$ corresponding to the given information sequence $U_0(D)$. The row distance indicates the minimum Euclidean distance between any two possible code sequences up to a certain length, and thus represents an upper bound to the minimum free Euclidean distance of the code. In order to make sure that the upper bound equals the exact free distance, we introduce another parameter called column distance. This parameter corresponds to the Euclidean distance between any two unmerged path of a certain length. The column distance $c_{U_0}(l)$ of order l from a given information sequence $U_0(D)$ is therefore defined by

$$c_{U_0}(l) = \min_{\substack{V(D)\neq V_{U_0(D)} \\ \deg V(D)\leq nl}} d_E(V(D), V_{U_0}(D)). \tag{4.22}$$

Since the column distance is the distance between any two code sequences truncated after l time units, it corresponds to the distance between the orthogonal projections of infinite code sequences onto an nl-dimensional space. This represents a

lower bound on the minimum free distance of the code. If the column distance equals the row distance for any l, then the free distance of the code equals their value. The difficulty of this approach is that l may be very large before the column distance and the row distance are equal. In general however, the row distance converges to the free distance much faster than the column distance. Therefore, it is possible to use the row distance in the search algorithm and verify with the code found that the row distance equals the column distance for some l.

The simulated annealing algorithm is a random search algorithm that optimizes a function of many parameters, which is not necessary simple to study analytically, or easily computable. In our case, the trellis code parameter that we are interested in optimizing is the minimum free squared Euclidean distance and the parameters to adjust are the $n2^{k+m}$ real number coefficients on the branches of the trellis. In [26], this algorithm was already used to search for binary convolutional codes, by changing randomly the binary coefficients of the generator at each iteration. In our case, a simulated algorithm was implemented as shown by the flow chart in Figure 4.13. For a given temperature, we perturbate the old coefficients to provoke the growth of the free distance. If the free distance increases, then we select these new coefficients, otherwise we select them with a probability depending on the new free distance and the temperature. After 100 loops, we decrease the temperature and start again.

The idea of this process is based on what happens during the crystalization of a liquid. If the temperature goes down too fast, then the crystal does not form, since atoms do not have time to position themselves on time. Here, if the temperature goes down too fast, we will quickly reach a situation where we only allow the growth of

the free distance. Assuming we have reached a local maximum for the free distance, we will never reach the global maximum of the free distance for this code rate and constraint length. Therefore, the simulated annealing algorithm allows us to slowly converge towards a good maximum for the free distance, without being sure, however that the resulting code is optimal. In particular, as the number of parameters increases, the temperature should go down slower to allow all parameters to adjust themselves to the best value.

However, three concepts resulting from the observation of the constructed codes strongly suggest that the codes found are optimal. First, the distance of the new codes is usually greater than the distance found with binary convolutional codes, which corresponds to our expectations. Second, for small constraint lengths, the free distance in most cases reaches the Heller upper bound [35] (see Chapter 3). Third, the best codes found exhibit certain properties that indicate that the codes are optimal:

- **Property 1:** A rate k/n real trellis code should be constructed using 2^m superimposed $n-$dimensional block codes, each with 2^k codewords. The block codewords determine the branch labels on the set of 2^k paths diverging from each state. This follows from the fact that the branches leaving each state have not yet remerged and can therefore be thought of as a rate k/n block code.

- **Property 2:** The block codes should be superimposed in such a way that the number of nearest neighbor code sequences in the trellis is maximized. This can be viewed as applying the philosophy of constructing

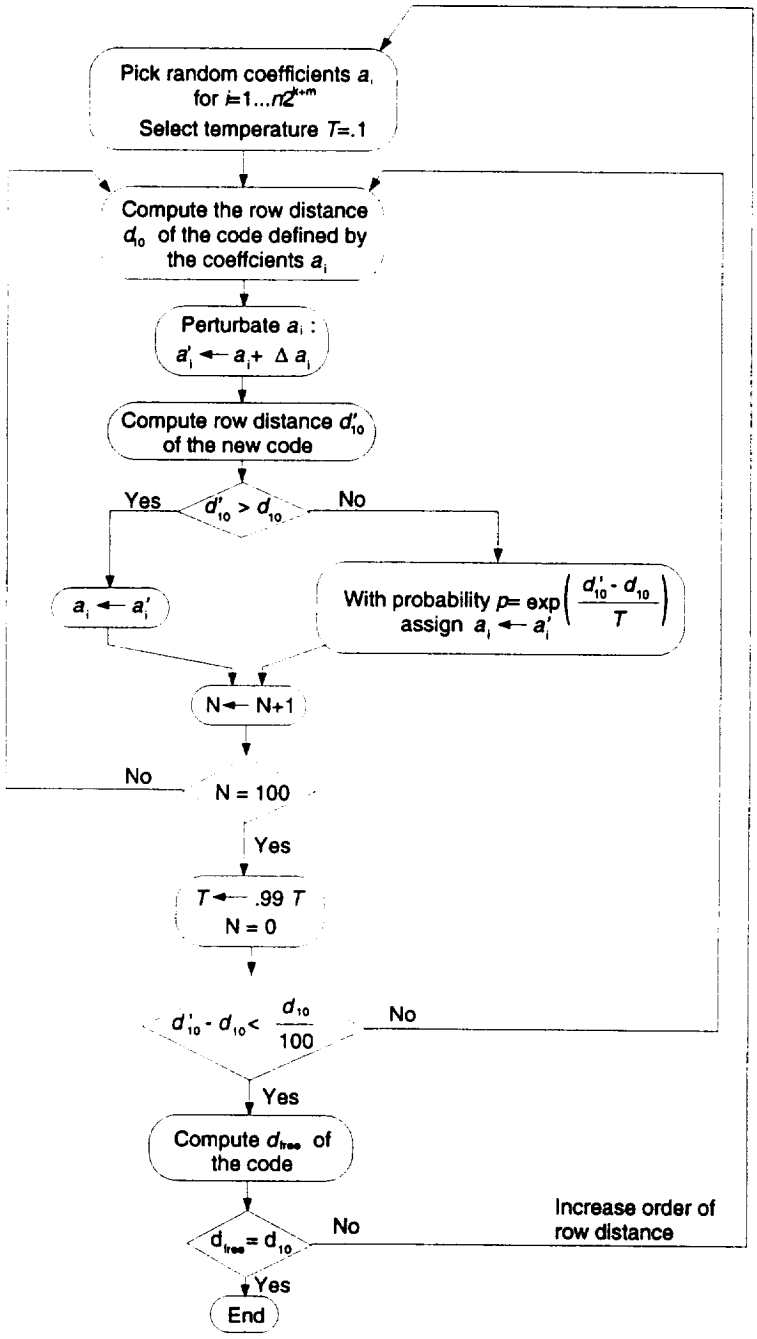

Figure 4.13: Simulated annealing algorithm to optimize a real number trellis code.

lattices with large density to the construction of infinite-dimensional block codes.

- **Property 3:** The lattice codes assigned to two states connected to the same state at the next time unit should be identical, thereby reducing the total number of lattice codes. This is due to the symmetry of the trellis.

Example 4.6.2 *Let us consider the 4-state rate 1/2 real number trellis code shown in Figure 4.10. Following rule 1, at each state, a block code with two codewords in two dimensions should be constructed. The best block code with two codewords in two dimensions can be constructed by positioning two opposite points on the energy 1 circle as shown in Figure 4.14. In the case of a binary block code, the points are restricted to the positions shown by the square. For the real case, the 4 block codes of two points are not restricted to these 4 positions. Using rule 2, we want to equate the distance separating the first two diverging and remerging paths of the trellis, that is, the path of length 3, and the path of length 4. This yields the following equation:*

$$d([a_0,a_1],[a_2,a_3])^2 + d([a_0,a_1],[a_4,a_5])^2 +$$

$$d([a_0,a_1],[a_8,a_9])^2 =$$

$$d([a_0,a_1],[a_2,a_3])^2 + d([a_0,a_1],[a_6,a_7])^2 +$$

$$d([a_0,a_1],[a_{12},a_{13}])^2 + d([a_0,a_1],[a_8,a_9])^2 \qquad (4.23)$$

Using rule 3, state 1 and 3 should use the same block code, and state 2 and 4 should

use the same block code. This simplifies (4.23) to

$$d([a_0, a_1], [a_4, a_5])^2 = d([a_0, a_1], [a_6, a_7])^2 + d([a_0, a_1], [a_6, a_7])^2 \tag{4.24}$$

Using the notation shown in Figure 4.14, (4.24) yields

$$D^2 = d^2 + d^2 \tag{4.25}$$

Assuming the usual normalization to energy 1, it is also possible to write

$$D^2 + d^2 = 4, \tag{4.26}$$

thus yielding

$$\begin{cases} d = \sqrt{\frac{2}{3}} \\ D = \sqrt{\frac{4}{3}}. \end{cases} \tag{4.27}$$

While for the best binary code, the squared Euclidean distance between the first diverging and remerging path and the all zero path is 10, and the distance between the second path and the all zero path is 12, they both equal 10.67 in the real number case. After computation, the minimum free squared distance of the real code is determined to be 10.67. This gain in distance corresponds to an asymptotic coding gain of .28 dB by using the real code as opposed to the binary code.

The previous example showed that by not restricting the points to the squared constellation (QPSK) used when transmitting binary convolutional code sequences, it is possible to increase the minimum free squared Euclidean distance of the code. This distance growth comes from the fact that diverging and remerging paths in a trellis topology do not have the same length. Thus, when constructing the code over the binary field, it is not possible to equate the distances between the all zero path

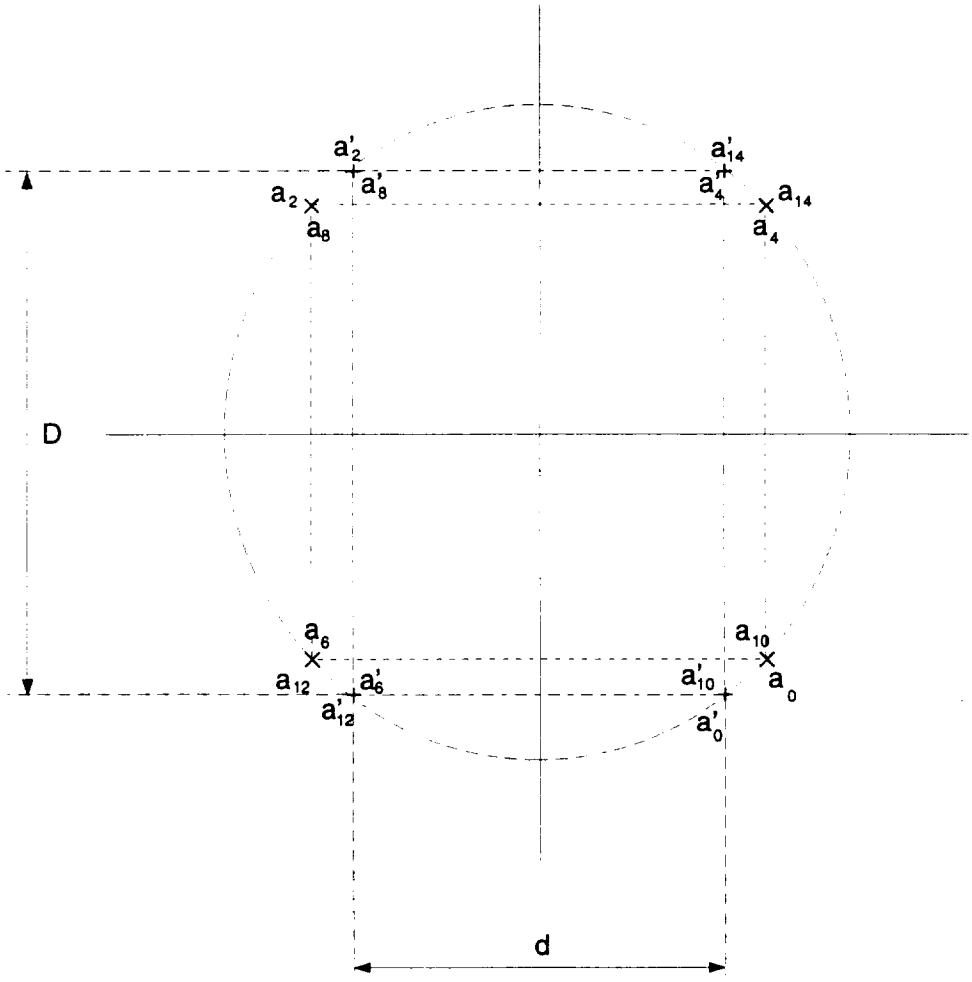

Figure 4.14: Binary and real constellations

and diverging paths of different length. Comparing this concept to the block code problem, it is the same idea than restricting points to the Z^n lattice instead of using the best sphere packing in n dimensions. The best lattice in n dimensions does most often correspond to the structure for which each point in the lattice has the largest number of nearest neighbors. For example, in two dimensions, the squared lattice has 4 nearest neighbors per point, while the hexagonal lattice has 6 nearest neighbors per point and a better packing density. Here, although we are constructing infinite dimensional sequences, it is also advantageous to increase the number of nearest neighbors to the all zero sequence, in order to increase the free distance of the code.

However, recall from Chapter 1 that the probability of error when using a trellis code on a binary symmetric channel with crossover probability p can be upper bounded by

$$P_b(E) \approx \frac{1}{k} B_{d_{free}} 2^{d_{free}} p^{d_{free}/2}, \tag{4.28}$$

where B_d is the total number of nonzero information bits on all weight d paths. Thus, increasing the number of nearest neighbors also increases B_d, and therefore the upper bound on the probability of error when p is sufficiently large, that is when the SNR is sufficiently low. The asymptotic coding gain, that is, at high SNR, indeed increases with the free distance. But at lower SNR, this may not be so advantageous. In order to have an idea of the improvement by using the real code described in example 4.6.2, we simulated the performance of both binary and real codes, and the result is shown in Figure 4.15.

Although the real number code has better free distance than the binary convo-

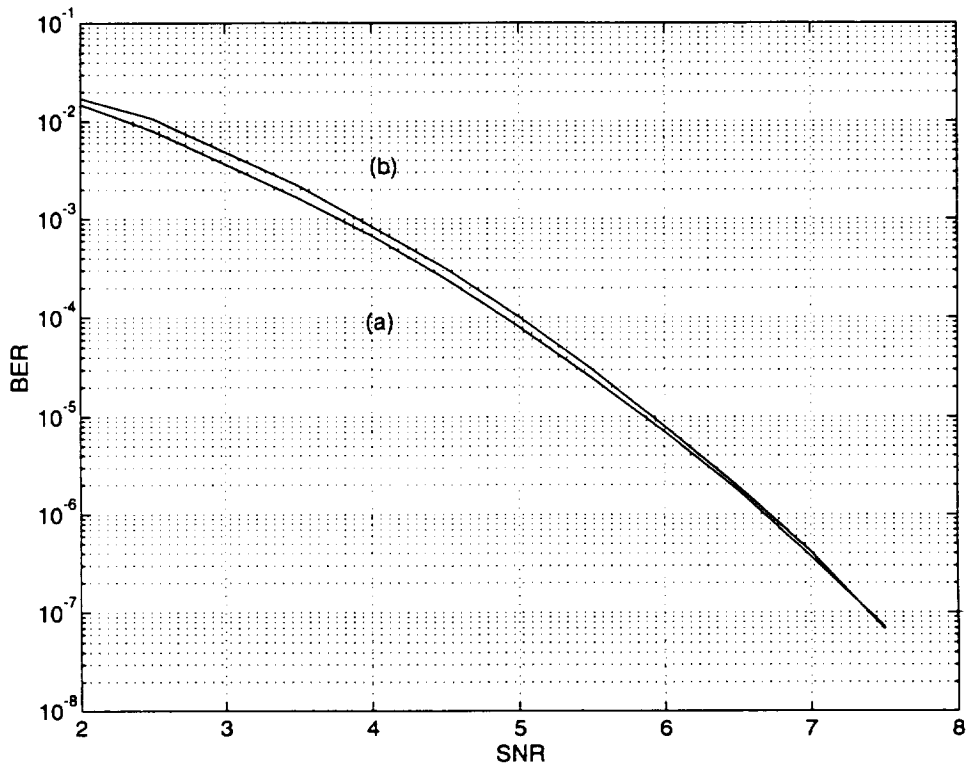

Figure 4.15: Simulation of the performance of the best 4 state, rate 1/2 trellis codes.

(a) binary, (b) real number

lutional code, the real code does not perform as well for low SNR than the binary code. As we just noted, this is due to the larger number of sequences separated by the minimum free distance in the real number code than the binary code. Therefore, the gain obtained from this growth of the free distance is only achieved for very large SNR. However, since the growth of the free distance is always achieved by increasing the number of nearest neighbors, we must realize that our goal depends on what SNR our channel is working at. For large SNR, we want to maximize the free distance, for lower SNR, we may rather want to optimize the number of nearest neighbors. However, the SNR affects the probability of error as a power of the minimum distance, whereas the number of nearest neighbors is only a multiplicative factor. It is therefore interesting to first select codes which optimize the free distance, and then select the code with the smallest number of nearest neighbor sequences.

4.6.3 Results

We limited our direct construction of real number trellis codes to relatively simple code rates, since the search time increases exponentially with k and m, and linearly with n, for the total number of parameters is $n2^{k+m}$. Therefore, we only constructed codes for which the trellis topology did not include any parallel branches, that is for codes with rates lower than 1. Codes with rates greater than 1 will be constructed in the next chapter, where we impose geometric uniformity, which reduces the search time. Table 4.2 shows the distance of our new codes along with the distance of the best binary convolutional codes of the same rate, the Heller upper bound [35] on the

real number codes distance, and the Griesmer upper bound [33] on the binary codes distance.

For most codes, the difference between the two bounds can be gained by constructing the code over the real numbers instead of the binary numbers. Figure 4.16 shows the Griesmer and Heller upper bounds along with our constructed codes. We can see that real number codes have a distance that reaches the Heller upper bound, thus gaining some distance over binary codes limited by the Griesmer upper bound. In the next chapter, we will construct larger constraint length codes and see that this improvement is the same for longer codes. As the constraint length increases, however, the difference between the Heller bound and the Griesmer bound tends to decrease, thus reducing the possibility of improving the distance by constructing codes over the real field.

Note also that since we normalized the average energy per symbol to 1, the minimum free distance obtained for code with same k and same m, *i.e.* same topology, have almost the same minimum free distance, whatever the dimensionality n is. This suggests that the topology of the trellis is responsible for the limit on the free distance rather than the dimensionality of each individual branch. In particular, whereas for block codes, increasing k and n while keeping the rate constant yields a minimum distance growth, the trellis code topology requires that we increase m along with k, rather than n in order to increase the minimum free distance per symbol.

A problem with real number trellis codes is that the concept of catastrophic encoder does not really exist since most codes do not have the same numbers on different branches, as binary catastrophic trellises do. However, some numbers can be very

m	rate	d_{free} (real codes)	d_{free} (bin. codes)	d_{free} (trellis codes)	Upper bound (real codes)	Upper bound (bin. codes)
1	1/1	8	4*	1/1 BPSK (4)	8	8
2	1/1	8	8	1/1 BPSK (8)	10.67	8
3	1/1	12	12	1/1 BPSK (12)	13.33	12
1	1/2	8†	6*	1/2 QPSK (6)	8	8
2	1/2	10.66	10	1/2 QPSK (10)	10.66	10
3	1/2	13.33	12	1/2 QPSK (12)	13.33	12
4	1/2	16	14*	1/2 QPSK (14)	16	16
1	1/3	8†	6.67	1/1 $(BPSK)^3$ (6.67)	8	8
2	1/3	10.66	10.66	1/3 $(BPSK)^3$ (10.66)	10.66	10.66
1	1/4	8	8	1/2 $(QPSK)^2$ (8)	8	8
2	1/4	10.5	10	1/2 $(QPSK)^2$ (10)	10.66	10
2	2/3	4.3	4	2/3 $(BPSK)^3$ (4)	5.33	5.33
2	2/2	3.45	2*	2/3 (8QAM) (3.2)	5.33	4

*: non-catastrophic codes.

†: asymptotically catastrophic

Average energy per symbol=1.

Table 4.2: Real number trellis codes constructed by a direct simulated annealing

close, so there can be long diverging unremerged paths with very low distance between them. Such a code would perform badly in terms of probability of information bit errors, since a small error in the code sequence could lead to a long incorrect path. Therefore, we call these codes *asymptotically catastrophic* in reference to a paper by Hemmati and Costello [52] in which they study binary convolutional encoders which have such long paths with low distance even though the encoder is not catastrophic. In Table 4.2, the asymptotically catastrophic codes are marked with a †.

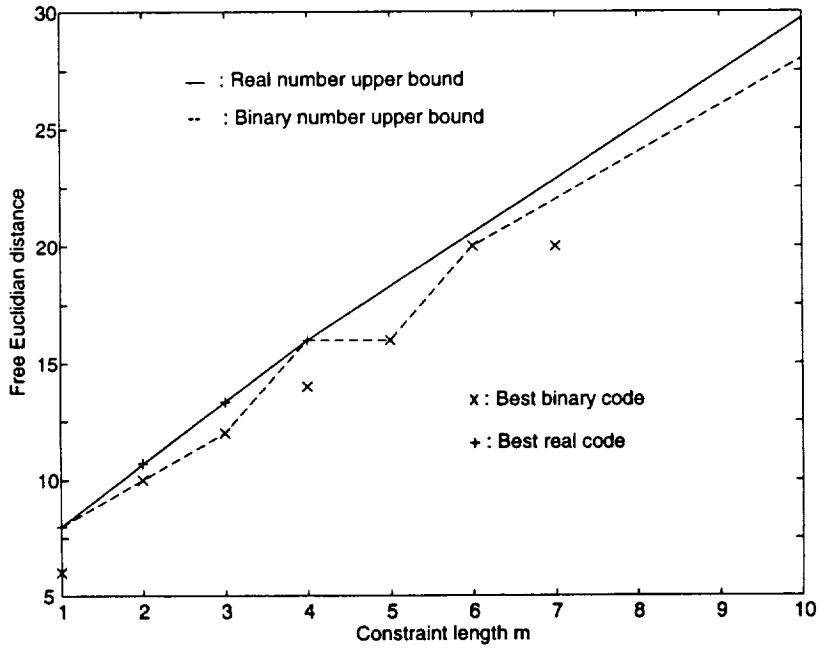

Figure 4.16: Rate 1/2 binary and real trellis code distances and upper bounds.

4.7 Conclusion

We have seen in this chapter different ways of constructing codes over the real field, and we noted an improvement for the minimum free distance of the codes corre-

sponding to the bounds derived in the previous chapter. Although our simulated annealing algorithm is very limited by its slow convergence due to the large number of parameters to adjust, we have been able to identify some real number codes that have a better free distance than binary codes of the same rate. However, in order to construct larger codes, it is necessary to reduce the number of parameters in our simulated annealing without reducing our chances to find better real number codes.

We can observe that some of the codes constructed have the property of being geometrically uniform, that is, have the same distance spectrum from any code sequence. This suggests that it may be possible to reduce our search to geometrically uniform codes only, and still obtain an improvement over binary codes. So far, there has not been any good technique to systematically construct geometrically uniform codes, since it is necessary to use an underlying convolutional code over a group as mentioned in Section 4.2.2. The next chapter describes a new approach based on a pure geometric construction, and we will see that the number of parameters to adjust by simulated annealing is dramatically reduced. This will allow us to construct much longer codes, and note further improvement of real number codes as opposed to binary codes, or trellis codes of rate greater than 1 constructed on usual constellations.

CHAPTER 5

A GEOMETRIC CONSTRUCTION OF GEOMETRICALLY UNIFORM TRELLIS CODES

5.1 Geometrically uniform trellis codes

The concept of using trellis codes over $\mathbb{R}^n$ for data transmission at high rates was first introduced by Ungerboeck [3]. The main idea was to decompose a trellis code into a binary convolutional encoder followed by a mapper that transforms binary codewords into modulated signals. In order to have a code with large minimum free squared Euclidean distance, set partitioning was used to maintain a fixed relationship between the minimum free Hamming distance of the underlying binary code and the minimum free squared Euclidean distance of the resulting trellis code.

Later on, the concept of geometric uniformity was introduced by Forney [15] in order to generalize the notion of linearity previously encountered with convolutional codes. It was noted that some trellis codes were geometrically uniform, even though the underlying convolutional code was not linear, such as Wei's eight state trellis code

[53] recommended in the CCITT V.32 standard. Further, Massey and Mittelholzer [54] found that geometrically uniform codes on PSK constellations can be constructed by using convolutional codes over rings such as Z_n. This concept was then generalized by Loeliger [16] and Trott [51], who developed the concept of isometry codes which relates the algebraic structure of a trellis code to the geometric properties of its signal set.

Earlier, Divsalar, Simon, and Yuen [55] tried to modify the constellations on which trellis codes are constructed, and obtained some improvement over regular constellations. Combining the idea of asymmetric constellations with the concept of geometric uniformity, Benedetto, Garello, Mondin, and Montorsi [56, 57] developed some trellis codes over multi-dimensional asymmetric constellations while imposing the geometric uniformity constraint in order to simplify the performance analysis.

All these approaches view trellis codes as q-ary convolutional codes mapped onto signal sets whose labels have been assigned using set partitioning techniques. This decomposition of the signal set labeling and the trellis code labeling makes it impossible to prove that the resulting trellis code is optimal. A first attempt by Calderbank and Mazo [58] was made to view the trellis code as a set of real labels directly assigned to the trellis, but did not lead to a systematic construction. In this dissertation, we introduce a new method of constructing geometrically uniform trellis codes based on a geometric construction of the trellis. The most important idea is that for a given trellis topology, the branches of the trellis which usually contain the generator of a convolutional code completely define the entire geometrically uniform trellis code, *i.e.*, $m + k$ branches are sufficient to define the entire trellis code, where 2^m is the

total number of states, and 2^k is the number of paths diverging from each state.

The usual technique for constructing trellis codes consists of decomposing a given signal constellation in order to adapt it to a trellis topology. Here, we decompose the trellis topology in order to find the best signal constellation for this topology. This technique allows us to directly search for the signal constellation and trellis code together which optimize the minimum free squared Euclidean distance between code sequences with a geometrically uniform constraint.

5.2 Trellis topology and generating branches

A trellis topology $T(k, p, m)$ is entirely determined by the number 2^k of branches leaving each state, the number 2^p of parallel branches leaving each state, and the total number of states 2^m. In this dissertation, we use binary types of trellis topologies, but they can be generalized to other structures such as ternary trellis topologies with 3^k branches, 3^p parallel branches, and 3^m states [59]. An example of a $T(3, 2, 2)$ binary topology is shown in Figure 5.1. Let us also define $\tilde{k} = k - p$, where $2^{\tilde{k}}$ is the number of sets of 2^p parallel branches leaving each state. A trellis code can then be constructed on this topology by assigning a symbol from $\mathbb{R}^n$ to each branch of the trellis structure. A binary input sequence then selects a path through the trellis structure, and the symbols along that path are transmitted. The rate R of the code is then given by

$$R = \frac{k}{n}. \tag{5.1}$$

Assuming that the branch labels are independent of time, there are $n2^{k+m}$ real numbers to assign in the trellis. These numbers must be assigned so that the minimum free squared Euclidean distance between any two diverging and remerging paths is maximized. Because of the large number of symbols to assign, it is desirable to reduce this number as much as possible while maintaining an optimum free distance. Even though it has not been proven that geometrically uniform codes are optimal, it has been shown that most of the best trellis codes that have been previously designed are geometrically uniform.

We will now show that by imposing geometric uniformity, only $n(m + k)$ real numbers need to be assigned in the trellis, and that each n-dimensional symbol must be taken from a certain type of constellation. First, we decompose the trellis topology in order to find the best signal constellation, and in the following section, we study which trellis branches must be specified in order to construct an entire trellis code.

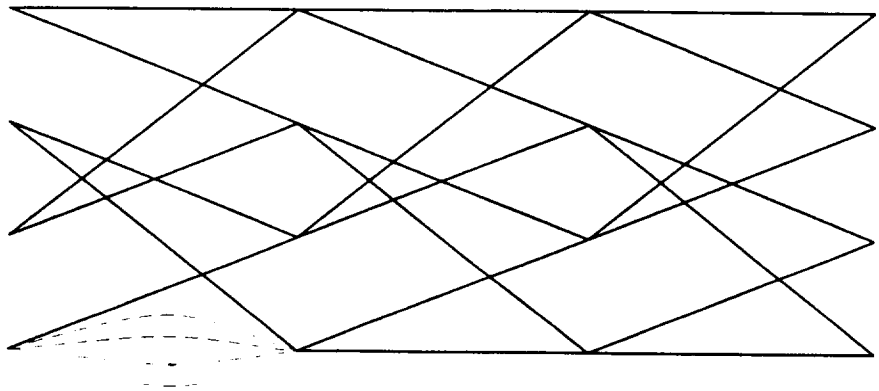

Figure 5.1: T(3,2,2) topology

5.2.1 Parallel branches

Parallel branches represent diverging and remerging paths of length 1. Thus, in order to optimize the minimum free squared Euclidean distance of a trellis code, we must optimize the minimum Euclidean distance among all n-dimensional symbols assigned to a set of parallel branches. This can be done using a block code with 2^p n-dimensional symbols. For a trellis code to be geometrically uniform even among its parallel branches, the set of block codewords must be geometrically uniform, that is, the set of distances from any codeword to all the others must be the same. This can be obtained by positioning the points uniformly on an n-dimensional sphere. However, such a strong condition limits the number of parallel branches, and thus the rate of the code. A weaker condition is to simply require that the *minimum* distance between any codeword and all the others is the same. This allows us to construct the block code by choosing the codewords from a uniform n-dimensional set of points within some boundary. This type of uniform set of points is called "geometrically uniform up to the boundary effects" by Forney [15]. Thus, there are two types of geometrically uniform sets of points:

- Perfectly geometrically uniform (PGU) set (which can be part of an infinite geometrically uniform set), see Figure 5.2 (a).
- Geometrically uniform up to boundary effects (BEGU) set (part of an infinite geometrically uniform set), see Figure 5.2 (b).

If p is less than or equal to n, a p dimensional hyperrectangle can always be constructed, thus yielding a perfectly geometrically uniform set of points. If $p > n$,

the set of points is either BEGU or PGU on an n-dimensional sphere.

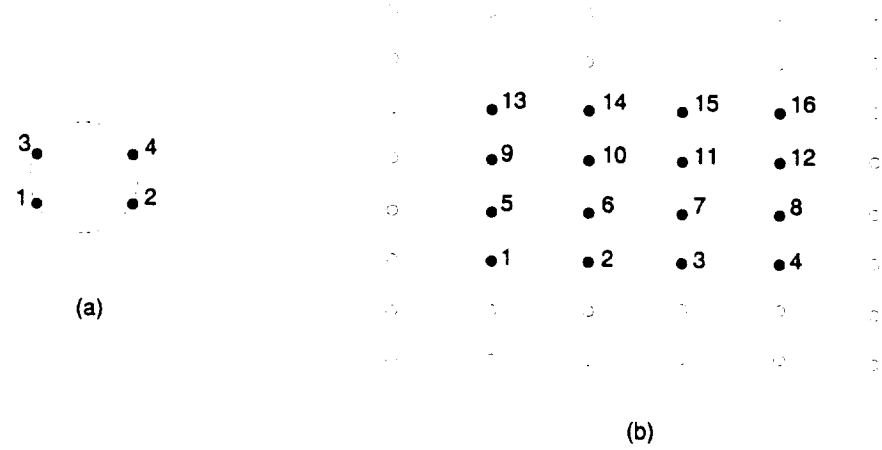

Figure 5.2: (a) Perfectly geometrically uniform set of points in 2 dimensions, (b) Geometrically uniform up to boundary effects set of points in 2 dimensions.

These two possible types of geometrically uniform parallel branch sets lead to two types of geometrically uniform trellis codes. The first type consists of a set of sequences for which the set of distances from any one sequence to all the others is the same (a PGU trellis code). The second type (a BEGU trellis code) consists of a set of sequences for which the *minimum* distance between any sequence and all the others is the same, and if the number of parallel branches is infinite, thus yielding an entire infinite geometrically uniform set of parallel branch points, the trellis code is PGU. This second type of trellis code is also called GU with respect to the trellis paths of length greater than 1.

For PGU trellis codes, a similar block code must be used on every set of parallel branches in the trellis. Therefore, once 2^p points have been assigned to one set of parallel branches, all the other parallel branch PB sets in the trellis are determined

by selecting a representative point, and then choosing the other points geometrically with respect to the representative point in the same way that the points were selected for the first set of parallel branches. For instance, for the PGU set in Figure 5.2 (a), suppose there are 2 parallel branches. Then, if (1,4) is selected as the first PB set, then for all other PB sets, only one branch must be chosen and the other one follows, that is, 1 implies 4, 4 implies 1, 2 implies 3, and 3 implies 2. Note that all PB sets constructed this way are similar, that is, constructed by orthogonal transformation from the first PB set.

For BEGU trellis codes, the other points must be selected by respecting a fixed minimum distance between them and the original set of points. For instance, for the PGU set in Figure 5.2 (b), suppose there are 4 parallel branches. Then, if (1,3,9,11) is selected as the first PB set, then all other PB sets are obtained by translation from the first PB set, that is, 2 implies (2,4,10,12), 5 implies (5,7,13,15), and 6 implies (6,8,14,16). Due to the boundary effects, the distance between two PB sets is defined by the minimum distance between any two points from each PB set. This means in particular that for each set of parallel branches, a different portion of the infinite GU structure can be taken. For energy purpose, the PB sets are then formed by taking the 2^p points with smallest energy in the infinite set.

5.2.2 Diverging branches from the same state

Leaving each state of the trellis are 2^k sets of parallel branches diverging to different states. They correspond to the orthogonal projection of all code sequences starting

from that state onto an n-dimensional space. From any state of the trellis, the sets of symbols on diverging branches must therefore be geometrically similar. Given one symbol on a parallel branch, the set of symbols on all other branches must be geometrically constructed in a similar fashion. Therefore, there must exist a set of diverging parallel branch (DPB) points that is PGU formed by selecting at least one symbol from each diverging PB set. Figure 5.3 shows four sets of points (a), (b), (c), and (d) constructed from the association of two PB sets.

In case (a), the PB set is PGU, thus, it is possible to construct a PGU set of diverging parallel branches (DPB set) by rotating the PB set around its center, by doing a symmetry around a hyperplane passing through the center, or by combining these two operations. In this case, the DPB set consists of all points from both PB sets. In case (b), the same PGU PB set is translated, thus, the PGU DPB set consists of only one point of each PB set corresponding to the extremities of the translation vector, such as (1,1') in the Figure. DPB sets constructed in both cases (a) and (b) can be used in the trellis shown in Figure 5.4. In case (c), the PB set is only BEGU, thus only translations are possible as shown in (c) of Figure 5.3 while maintaining the geometric uniformity of the association of both sets. Therefore, the DPB set consists also of one point from each PB set as in case (b). Finally, in case (d), the PGU PB set has strictly more than 2^n points, thus a translation does not lead to a GU set of points. A rotation as in (a) would yield a small distance between points, thus making the resulting trellis code "bad" in terms of minimum Euclidean free distance. Hence, if $p > n$, the PB set must not be PGU but rather BEGU. This is due to the fact that a PGU PB set with less than 2^n points can also be seen as part of an infinite periodic

GU structure, whereas a PGU PB set with strictly more than 2^n points cannot be periodically translated to form a GU structure.

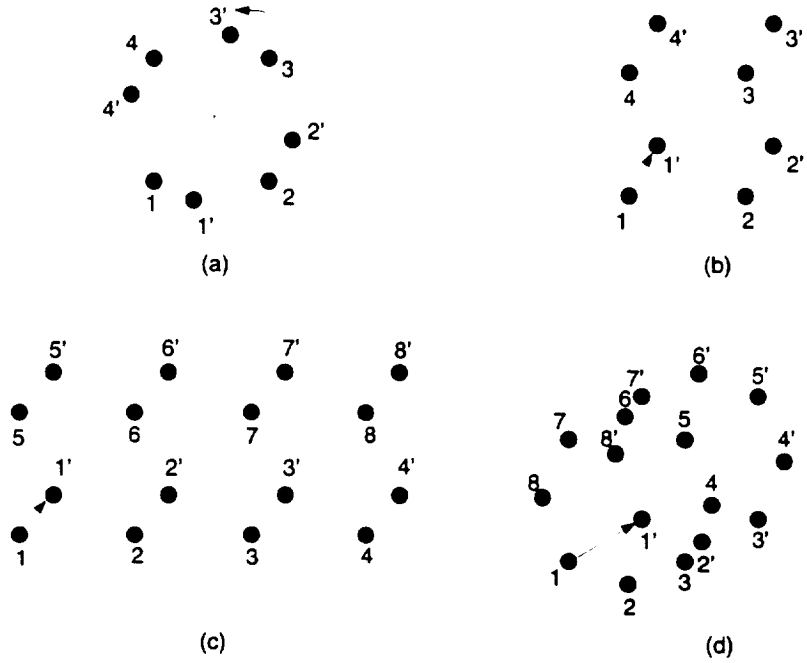

Figure 5.3: (a) PGU set of diverging PGU parallel branches, (b) BEGU set of diverging PGU parallel branches, (c) BEGU set of diverging BEGU parallel branches, (d) Non GU translated sets of PGU parallel branches.

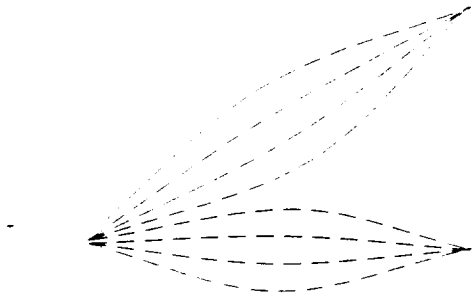

Figure 5.4: Two diverging branches with four parallel branches at a trellis state.

In that example, we only considered two diverging sets of parallel branches. More generally, the set of translated points or rotated points must form a PGU set of points

such that from any point of the diverging parallel branches, the same set of points can be geometrically selected. An example is shown in Figure 5.5 for 4 diverging sets of 4 PGU parallel branches (a), 4 diverging sets of 2 PGU parallel branches (b), and 4 diverging sets of 8 BEGU parallel branches (c). Therefore, although the entire constellation may be only BEGU, it is necessary to construct PGU DPB sets.

A catastrophic trellis is a trellis for which there exists long unmerged paths separated by a small distance. It is equivalent to the concept of catastrophic encoders for binary codes, for which there exists long unmerged paths separated by no distance at all, except on the first branches. Note that for binary encoders, it is possible to use an equivalent minimal noncatastrophic encoder which yields a code with the same distance but on finite length paths. For real number trellis codes, this is equivalent to label the trellis differently, thus yielding the same code with small minimum free Euclidean distance. To avoid catastrophic trellises or codes with very small distance, it is necessary to have at least two different geometrically similar DPB sets, for otherwise there would exist different paths having the same symbols on their branches. The signal constellation must therefore be constructed by selecting at least two geometrically similar PGU DPB sets, and then associating PB sets with each point of the DPB sets such that the whole constellation is BEGU. However, to keep the geometric uniformity, this second diverging branches set of point must be geometrically identical to the first one, that is, be a translation, rotation (if keeping the BEGU structure) or reflection of the first set. Note for example that Figure 5.5 (a) corresponds to the situation found in the trellis shown in Figure 5.1, since $(1, 2, 3, 4)$ and $(1''', 2''', 3''', 4''')$ correspond to the first two PB sets. The reflection of the DPB set consisting of points

1 and 1"' constitutes the second DPB set (1',1") for which the associated PB sets are $(1'', 2'', 3'', 4'')$ and $(1', 2', 3', 4')$.

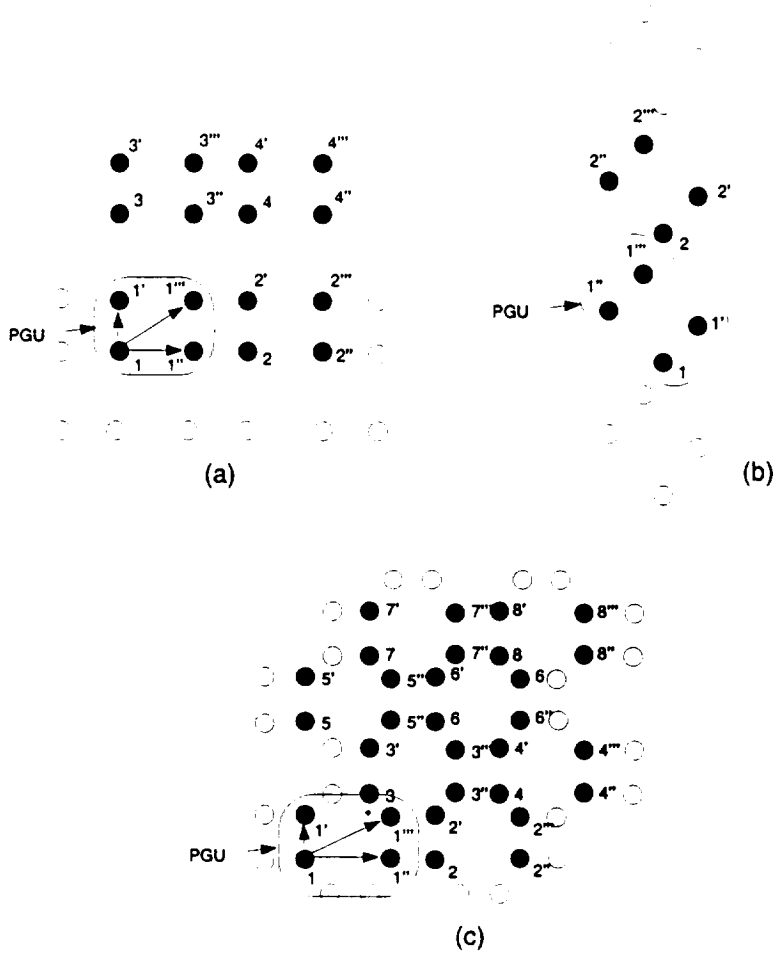

Figure 5.5: (a) 4 diverging sets of 4 parallel branches, (b) 4 diverging sets of 2 parallel branches, (c) 4 diverging sets of 8 parallel branches.

In conclusion, the signal constellation must therefore be constructed by selecting at least two geometrically similar PGU DPB sets, and then associating PB sets with each point of the DPB sets such that the whole constellation is BEGU. In order to construct geometrically similar PGU DPB sets, we can select 2^{n-k+p} geometrically uniform sets of 2^{k-p} points by (1) set-partitioning an n-dimensional hyperrectangle

into geometrically uniform sets of 2^{k-p} points if $(k-p) < n$ (see Figure 5.6 (a)), or (2) translating a $(k-p)$-dimensional hyperrectangle if $k-p = n$ (see Figure 5.6 (b)). In case (2), the two DPB sets together do not form a PGU set, but rather a BEGU set, and the associated PB points must be chosen to maintain the BEGU structure. One way to obtain a PGU set in case (2) is to rotate one DPB set with respect to the other, but this does not allow any flexibility in constructing sets of parallel branches. In other words, all points in the constellation would have to be on an n-dimensional sphere, which may lead to a poor minimum distance if the number of points is too large. Thus, only codes with $k = n$ can be constructed in this way (see Figure 5.6 (c)). In any case, if $\tilde{k} > n$, it is impossible to construct two PGU DPB sets from an n-dimensional hyperrectangle, thus forcing us to use rotation, which leads to codes with poor minimum distance since $k > n$. Therefore, no good geometrically uniform code exists for $\tilde{k} > n$. (See construction algorithm in Figure 5.13.) Note that usual trellis code constructions from Ungerboeck [3] or Wei [48] are never constructed on trellis topologies with $\tilde{k} > n$.

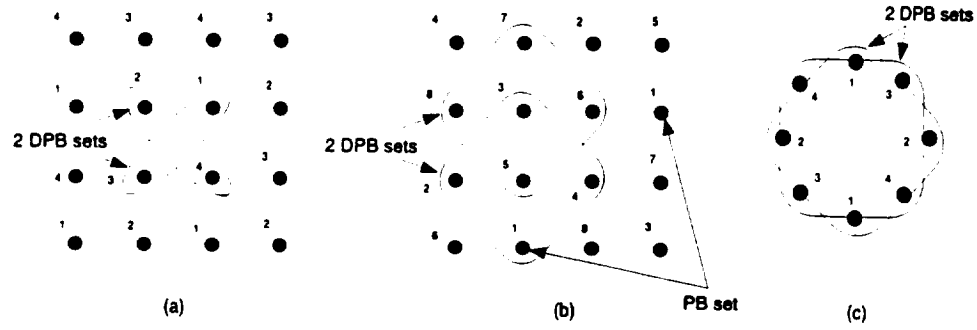

Figure 5.6: Constructing geometrically similar PGU DPB sets: (a) $\tilde{k} < n$ case, (b) $\tilde{k} = n$ case, (c) $k = n$ case.

5.2.3 Generating branches of the trellis

We have seen that every point of the signal set can be described as its position in the PB set and its position in the two or more DPB sets. Therefore, any symbol of the code can be described by the set of two indices (a, b), with $1 \leq a \leq l \cdot 2^{\tilde{k}}$ and $1 \leq b \leq 2^p$, where $2 \leq l \leq 2^{n-\tilde{k}}$. Due to the geometric uniformity constraint, only the first point of each diverging branches set needs to be specified, since the others are geometrically determined. Using this technique, only certain branches of the trellis must be specified, and the rest of the trellis can be completed using a geometric construction. To each DPB point is associated the PB set consisting of the set of 2^p points with lowest energy in an infinite BEGU set containing this DPB point. Without loss of generality, the lowest branch of the trellis can always be assigned a DPB point which serves as a reference point. Then, the set of diverging branches can be specified by the symbols on branches numbered 2^{p+j} for $0 \leq j \leq k - p + 1$, since all the other branches can then be assigned geometrically. Similarly, once the first set of diverging branches is specified, only the first branch of DPB sets starting from states numbered 2^k for $0 \leq k \leq m - 1$ must be specified, since the other sets can be constructed geometrically. Figure 5.7 shows for a $T(3, 2, 2)$ topology all the generating branches that must be specified in order to construct the entire trellis code. We note indeed that the specification of branch 4 completely defines the set of symbols on branches 5, 6, and 7. Similarly, once branches 0 to 7 have been assigned, the specification of branch 8 completely defines branches 9 to 15, and finally, once branches 0 to 15 have been assigned, branch 16 defines all branches from 17 to 31. For

each new generating branch, we can assign the set of other branches by maintaining the same set of distances from the new generating branch as the set of distances between points on lower branches and the branch 0 reference point.

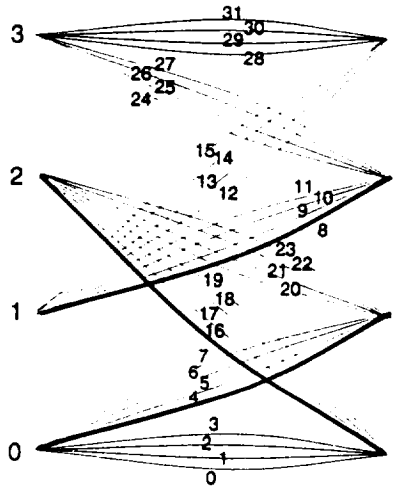

Figure 5.7: Generating branches of a geometrically uniform trellis code constructed on a $T(3,2,2)$.

The generating branches correspond to the branches usually assigned by the generator of a convolutional code based on a similar topology without the parallel branches. In the example shown in Figure 5.7, these branches correspond indeed to the path described by the information sequence (1,0,0) where the generator bits can be found for a 4 state rate $1/n$ convolutional code. For topologies with $\tilde{k} > 1$, the generating paths correspond to all information sequences starting with a first k-tuple composed of all zeros except in one position, and then as many zeros as needed to make the path remerge to the all zero state (delay). For instance, Figure 5.8 shows the generating branches of a $T(2,0,3)$ topology. Note that the first sequence has delay 2, and the second has delay 1 only, which correspond to a total of $m = 3$ delays. In general, the

number of branches to assign is therefore $m + k$. This unifies the theory of convolutional codes and trellis codes, since the search for the best trellis codes requires the same complexity than for convolutional codes. However, the free Euclidean distance does not only depend on the generator but also on the constellation. The next section explains how the optimization of both constellations and generators is achieved.

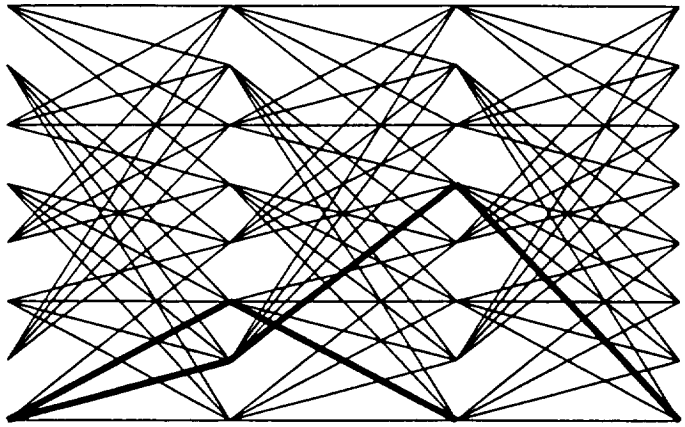

Figure 5.8: Generating branches of a geometrically uniform trellis code constructed on a $T(2, 0, 3)$.

5.3 Optimization of the free distance

Although PB sets represent block codes in n dimensions, and points must therefore be positioned in n-dimensional space in order to optimize the minimum distance between them, the DPB sets must not necessary be positioned in order to optimize the minimum distance between them. Specifically, diverging and remerging paths are not all of the same length, thus creating sequences of different dimensionalities.

5.3.1 Constellation optimization

In a finite dimensional space, the sphere packing problem shows that increasing the minimum distance between points usually increases the number of nearest neighbors to each point, that is, the number of points separated by the same distance. For trellis codes, this concept is the same if we consider the first remerged path which are separated by a finite number of dimensions. Although the number of dimensions that separate sequences in a trellis code varies, the distance between them can increase by increasing the number of nearest neighbor, that is, be separating different sequences by the same distance.

For example, for the 4 state rate 1/2 trellis code based on a $T(1,0,2)$ topology, the first two diverging and remerging paths are of dimensionality 3 and 4. The optimal convolutional code, which corresponds to a GU trellis code constructed on a squared PGU DPB set, has minimum free squared Euclidean distance 10. However, the first remerging path is separated from the reference path by a distance 10, while the second remerging path is separated by a distance 12. By changing the shape of the PGU DPB set to a rectangle, for which the short side is assigned to some of the branches of the dimension 4 sequence, and the long side is assigned to some of the branches of the dimension 3 sequence, it is possible to equate the distances between these two sequences and the reference sequence. Thus, both sequences are then separated by a squared distance of 10.667, which becomes the minimum free squared Euclidean distance of the modified code. Thus, one approach to optimizing the free distance of geometrically uniform trellis codes is to construct n-dimensional hyperrectangles,

instead of hypercubes, and to equate the distance between paths of different lengths. For the previous example, the shape of the rectangle can be optimized analytically. For more complicated codes, the length of the rectangle sides can be optimized by an optimization algorithm on n parameters, such as a simulated annealing [26].

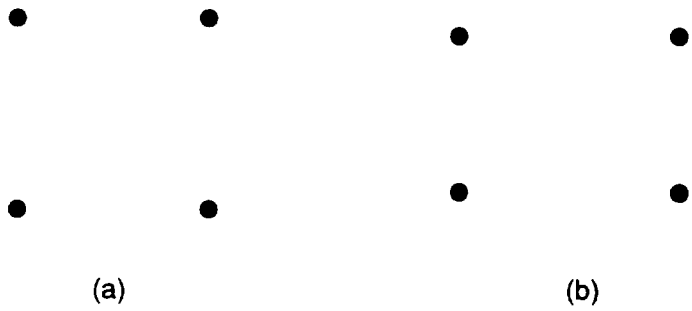

Figure 5.9: (a) Squared constellation of the 4 state rate 1/2 convolutional code, (b) Rectangular constellation for the same code.

This approach can also be applied to equating the distance between the parallel branches and the shortest diverging and remerging paths. In particular, by expanding the PGU DPB hyperrectangle, it is possible to also expand the PB sets, thus increasing the minimum distance between parallel branches and decreasing the minimum distance between diverging and remerging paths. For example, the 4 state rate 3/2 trellis code based on the $T(3,2,2)$ topology constructed by Ungerboeck [3] using a 16QAM constellation has a minimum distance between parallel branches of 1.6, whereas the minimum distance between the shortest diverging path and the reference path is 2.0. Thus, by expanding the PGU DPB square, it is possible to make both of these distances equal to 1.66, which becomes the minimum free squared Euclidean distance of the modified code (see Figure 5.10 (a)).

Note that this topology without parallel branches is the same as in the previous

example for the rate 1/2 code. Therefore, by going to a rectangular PGU DPB set, it is possible to equate the distances between the first remerging paths of dimensions 3 and 4. For the square PGU DPB set, the distances are 1.66 for the first path and 2.02 for the second path. However, for the rectangular PGU DPB set, the distance between both paths and the reference path is 1.68 (see Figure 5.10 (b)), which is the best minimum distance achievable with this topology.

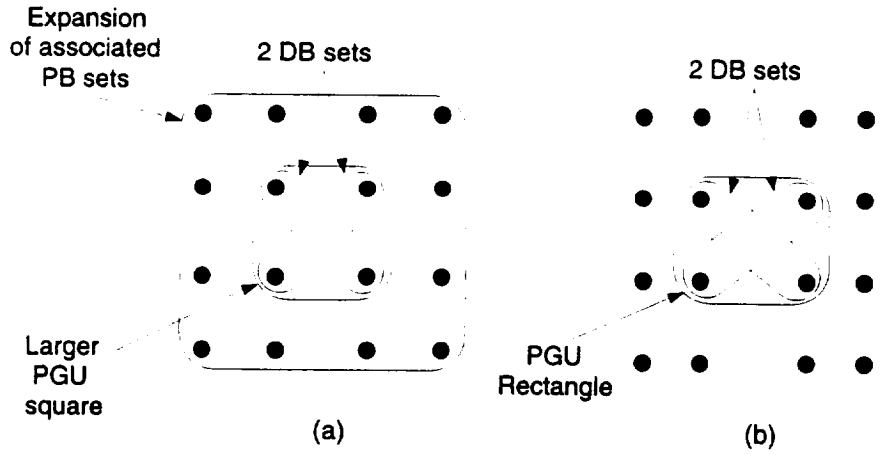

Figure 5.10: (a) Optimized squared PGU DPB set, (b) Optimized rectangular PGU DPB set.

Finally, it is possible to rotate PB sets if they are PGU in order to construct DPB sets. This can be advantageous as opposed to translating sets, since it requires less energy, and therefore points can be positioned further apart, thus yielding a greater minimum distance. For example, the 4 state, rate 2/2 trellis code based on the $T(2,1,2)$ topology constructed by Ungerboeck [3] using an 8AMPM (translated DPB sets) constellation has minimum free squared Euclidean distance 3.2. For that code, the PB set is a PGU segment of 2 points, and the first DPB set is constructed by rotating this segment by $\pi/2$ radians, thus yielding another PGU 4 points square

structure. The second DPB set is then obtained by translating the first set (See Figure 5.11 (a)). This translation can be as small as possible, and can asymptotically make the distance go to 4.0 as the translation magnitude decreases. However, if the two diverging branches sets merge into one (See Figure 5.11 (b)), the code becomes catastrophic. Therefore, another way of obtaining such a distance is to rotate the first DPB set to obtain the second one (See Figure 5.11 (c)), which also yields a code with Euclidean distance 4.0 while not being catastrophic. The angle of rotation does not change the code distance but the second set must be as distant as possible from the first set in order to make it as different as possible from a catastrophic code, thus yielding the 8PSK PGU constellation.

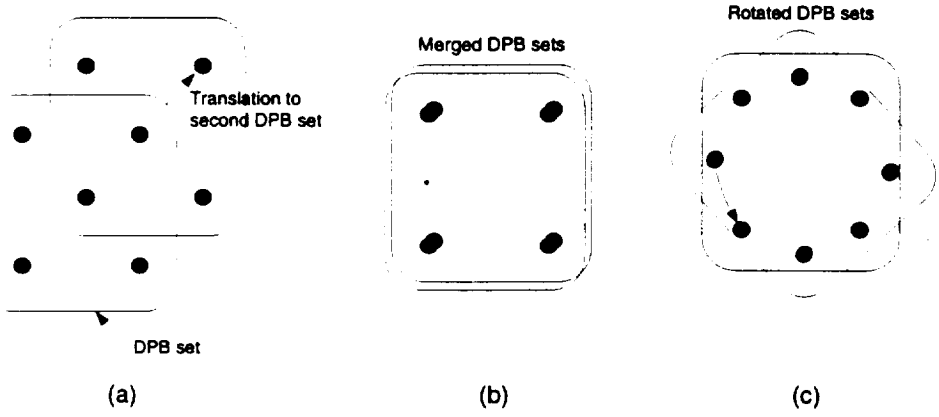

Figure 5.11: (a) 8AMPM constellation, (b) Merged 8AMPM constellation, (c) 8PSK constellation.

5.3.2 Generator optimization

At this point, we have only studied necessary conditions on the constellation for the trellis code to be geometrically uniform, and we have exhibited the generating

branches of a geometrically uniform trellis code. In this section, we study whether these conditions are also sufficient for the trellis code to be geometrically uniform, and it appears that not all possible generators lead to geometrically uniform codes when the constellation is only BEGU. For PGU constellations such as n-dimensional hyperrectangles or n-dimensional PSK, we can indeed select any PGU PB set, and at least two PGU similar DPB sets. However, for BEGU constellations, other restrictions have to be applied on the generating branches to obtain a geometric uniform trellis code.

- The first two states contain two different PGU DPB sets.
- The last *remerging* branches of the $\tilde{k}$ generating paths all merge to the bottom state, and must therefore all belong to the first PGU DPB set.

Taking into account these extra rules, the geometric generator matrix of a trellis code can be written as the generator matrix of a convolutional code, by listing in the $\tilde{k}$ rows of the matrix all the DPB points situated on each $\tilde{k}$ generating branches. Therefore, the matrix associated to a rate k/n code constructed on a $T(k, p, m)$

topology can be written as

$$G \triangleq \begin{bmatrix} a_1^1 & a_1^2 & \dots & a_1^{d+1} \\ a_2^1 & a_2^2 & \dots & a_2^{d+1} \\ \vdots & & & \vdots \\ a_i^1 & a_i^2 & \dots & a_i^{d+1} \\ a_{i+1}^1 & a_{i+1}^2 & \dots & a_{i+1}^d \\ \vdots & & & \vdots \\ a_{\tilde{k}}^1 & a_{\tilde{k}}^2 & \dots & a_{\tilde{k}}^d \end{bmatrix}, \tag{5.2}$$

where the delay $d = \lceil \frac{m}{\tilde{k}} \rceil$ and the number of branches with delay $d+1$ is $i = m+\tilde{k}-d\tilde{k}$.

For example, the generator matrix of the best 4 state rate 3/2 trellis code based on a $T(3,2,2)$ topology is given by

$$G = \begin{bmatrix} 4 & 3 & 4 \\ . & & \end{bmatrix} \tag{5.3}$$

and corresponds to the trellis and constellation shown in Figure 5.12.

This matrix format differs slightly from the generator matrix of a convolutional code, since the n dimensions are regrouped into one integer index between 1 and 2^n. Therefore, the matrix really corresponds to one column of a convolutional code generator regrouping the n dimensions for each delay. For example, the convolutional code matrix $\begin{bmatrix} 111 & 101 & 100 \end{bmatrix}$ corresponds to the trellis code generating matrix $\begin{bmatrix} 8 & 4 & 6 \end{bmatrix}$. Hence, the search for the best generator requires the same combinatorics as a convolutional code generator search and the free distance can be computed by using the Bahl and Larsen's algorithm to compute the free distance [43], since the code is geometrically uniform and therefore exhibits the same distance spectrum from

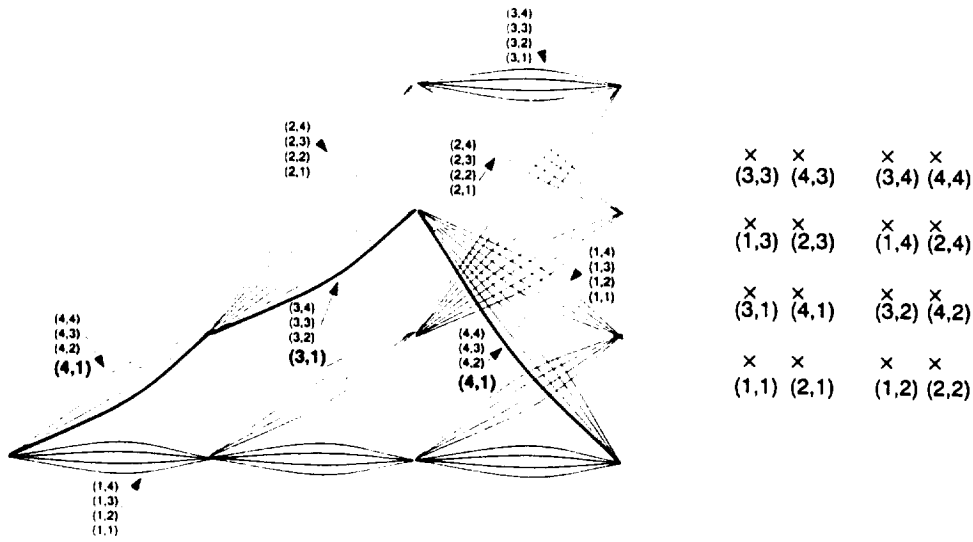

Figure 5.12: Trellis and constellation of the best 4 state rate 3/2 trellis code.

the all zero path than from any other path. However, if the constellation is BEGU only, the distance between two points from two different BEGU or PGU PB sets must be calculated by taking the minimum distance between the two sets as mentioned in Section 5.2.1.

5.4 Search for the best geometrically uniform code

Given all these rules to optimize constellation and generator, an algorithm to construct optimal geometrically uniform codes must include an iterative procedure to find the best constellation and generator on this constellation. The diagram of the search algorithm for finding the best trellis code for a given rate and topology is shown in Figure 5.13.

The idea of the algorithm is to base the constellation on the largest hyperrectangle that can be built in this dimension. The first test checks whether it is possible

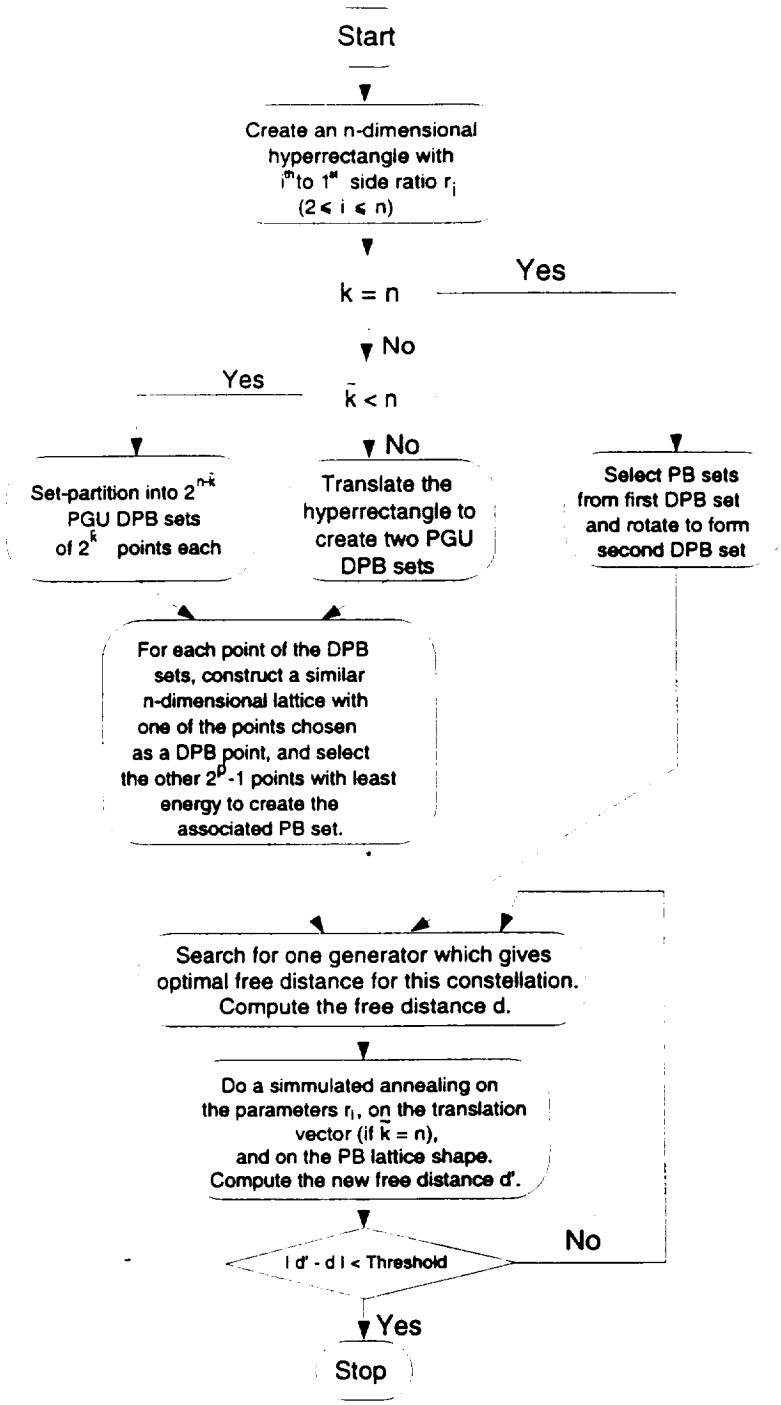

Figure 5.13: Search algorithm flow chart.

to construct the entire constellation on an n-dimensional sphere. For instance rate 2/2 codes can be constructed on PSK constellations ($k = n$) (See Figure 5.6 (c)). The second test checks whether the DPB sets can be constructed by selecting $2^{n-\tilde{k}}$ similar $\tilde{k}$-dimensional hyperrectangles, or if it is necessary to translate the first DPB set constructed on an n-dimensional hyperrectangle. For instance, if $k < n$ and $p = 0$ as for a convolutional code, we can always construct 2^{n-k} similar k-dimensional hypersquares within the n-dimensional hypersquare, by taking for instance the two diagonals of a square if $n = 2$ (See Figure 5.6 (a)). On the other hand, if $\tilde{k} = n$ such as for the 3/2 trellis code based on $T(3, 1, 3)$, it is necessary to translate the first square to obtain the two DPB sets, thus yielding a 16 QPSK-type constellation as shown in Figure 5.6 (b). Note that in this case, the PB lattices have to be constructed in order to maintain the BEGU structure started with the translated DPB sets. See Section 5.6 for more details.

Although the same constellation is used for all Ungerboeck's codes of the same rate, they correspond in fact to a different construction when p varies. For example, the rate 3/2 code with $m = 2$ is constructed with $p = 2$, thus leading to $\tilde{k} < n$. For $m = 3$, $p = 1$, hence, $\tilde{k} = n$. Therefore, the 16QAM constellation is decomposed into different DPB sets as shown in Figure 5.14. In any case, $p = 0$ would yield $\tilde{k} > n$, which cannot lead to a geometrically uniform code.

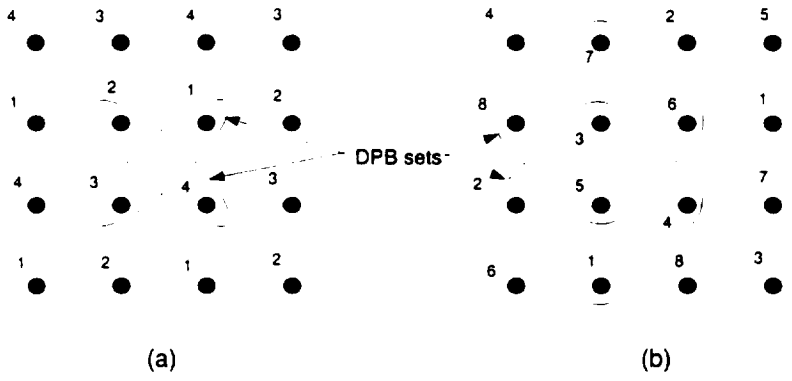

Figure 5.14: (a) 4 state rate 3/2 code ($p = 2$), (b) 8 state rate 3/2 code ($p = 1$).

5.5 Analysis of some existing geometrically uniform codes

To illustrate the theory explained in this dissertation, it is interesting to study some codes which are geometrically uniform even though they are constructed with a nonlinear convolutional code, as well as some non geometrically uniform codes.

5.5.1 Geometric uniformity of Wei's 8 state, rate 4/2 trellis code

In this section, we study the example of Wei's eight state trellis code [53] specified in the CCITT V.32 standard, which is based on a nonlinear convolutional code, but happens to be geometrically uniform. With our geometric construction, the generating branches automatically lead to the same code. Figure 5.15 shows the trellis diagram and the constellation associated to the trellis. On the constellation we identify the two PGU DPB sets, as well as their corresponding PB sets.

First, note that some states of the trellis as it was represented in [53] have been switched to obtain our usual representation of the $T(4, 2, 3)$ topology. Following the

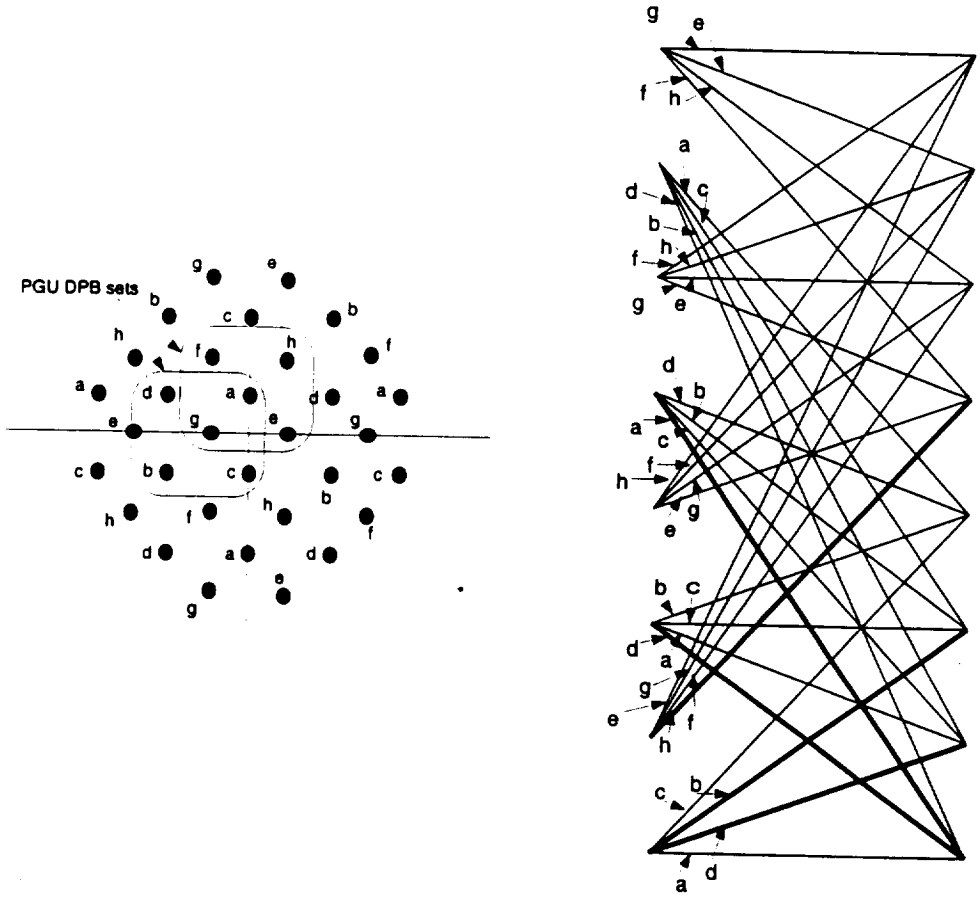

Figure 5.15:. Wei's eight state rate 4/2 trellis code on $T(4,2,3)$.

search algorithm flow chart, it is possible to identify this code as being geometrically uniform. Indeed, since $k \neq n$ and $\tilde{k} = n$, we must translate two squares which correspond to the two PGU DPB sets. The PB sets associated to each point of the DPB sets are such that the energy is minimized. The generator matrix of the code can be written

$$G = \begin{bmatrix} d & h & a \\ & & \\ b & d & \end{bmatrix}. \tag{5.4}$$

Note that the geometric position of $[a, d, b, c]$ is equivalent to $[a, c, b, d]$, hence, other equivalent geometrically uniform codes could be constructed from the same generator matrix. However, if $[a, d, b, c]$ was a rectangle instead of a square, they would not be equivalent anymore, and Wei's code would fail to be geometrically uniform, since the generator matrix above would produce interversion of some labels as opposed to Wei's code. This example shows that it is not always possible to encode a geometrically uniform code by using a linear convolutional code followed by a mapper, since not all orthogonal transformations from one DPB set to another can be expressed by a linear transformation in the binary field.

5.5.2 Non geometric uniformity of Wei's 64 state 5/4 code

It has been noted that all Wei's 4D trellis codes with strictly less than 32 states are geometrically uniform, whereas no 64 state code has been found geometrically uniform yet. This can be explained by the fact that all 4D codes with less than 32 states have been constructed on topologies with $\tilde{k} = 3 < n = 4$, whereas 64 state codes have been constructed using a topology with $\tilde{k} = 5 - 1 = n = 4$. For this code, Wei used a

decomposed 4D constellation into 64 subsets obtained by selecting 8 subsets in each composing 2D constellation. Thus, 3 bits select a point in the first 2D constellation and 3 bits select a point in the other 2D constellation. In order to understand the problem better, it is advantageous to compare it to the construction of a rate 3/2 code on the Cartesian product of two 1D constellation as shown in Figure 5.16. We then need 16 subsets obtained by selecting 4 subsets in each 1D constellation. Thus, 2 bits select a point in the first dimension and 2 bits in the other dimension. However, in order to construct two translated squares as shown in Figure 5.16, at least one bit must be different in each dimension to represent each DPB set. However, since there are only three bits on each branch of the underlying convolutional code, the fourth bit being added for a parallel branch, it is impossible to obtain the DPB sets as shown in Figure 5.16. Therefore, the separate labeling of each constituent subdimensional constellation does not allow one to construct a geometrically uniform code when $\tilde{k} = n$. One way of obtaining a geometrically uniform code would then be to either use a rate 2/4 underlying convolutional code, or authorize a direct labeling of the whole 4D constellation. However, with the geometric construction described previously, it is possible to construct a 64 state rate 5/4 trellis code with $p = 1$ which is geometrically uniform.

5.6 Construction of the best geometrically uniform codes for a trellis topology

Figure 5.13 shows the algorithm for constructing geometrically uniform trellis codes. In the preceding section, this algorithm was used to study the geometric uniformity

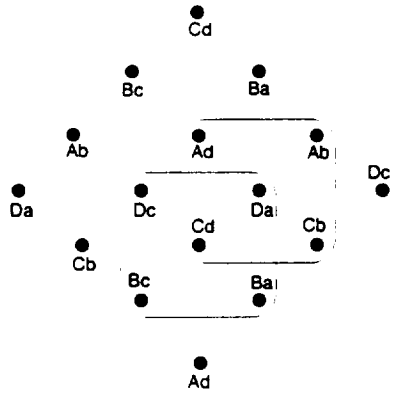

Figure 5.16: Cartesian product of two 1D constellations.

of some previously constructed trellis codes. In this section, we describe the implementation of the algorithm, and give a table of codes reaching a greater free distance than previously constructed trellis codes.

5.6.1 Implementation of the algorithm

The algorithm is divided into three main algorithms for constructing the signal constellation depending on the parameters k, p and n, a generator search, and a simulated annealing on the parameters specifying the constellation. First, we must construct an n-dimensional hyperrectangle with i^{th} to first side ratio r_i for $2 \leq i \leq n$. These $n-1$ parameters equal 1 for a hypercube. The construction is executed by layers. The first layer consists of two points separated by a distance 1, then by induction, the $i+1^{th}$ layer is constructed by copying the previous layers in a new dimension distant by r_{i+1} from the previous one. In other words, the coordinates X of any point of the rectangle can be put in the form

$$X = R \cdot B, \tag{5.5}$$

where $R = (1, r_2, \ldots, r_n)$ and $B = (b_1, b_2, \ldots, b_n)'$ with $b_i \in \{0, 1\}$, $1 \leq i \leq n$. Figure 5.17 shows a 3-dimensional hyperrectangle. An n-dimensional parameter d indicating the displacement of the center of the rectangle (not necessarily the zero energy point) is introduced for the simulated annealing.

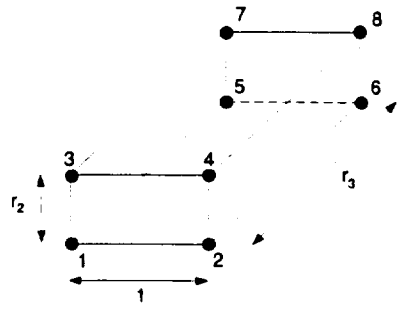

Figure 5.17: 3-dimensional hyperrectangle.

Geometrically uniform set partitioning of the main hyperrectangle

We now present the constructing algorithm associated to cases for which $k \neq n$ and $\tilde{k} < n$. The idea is to set-partition the hyperrectangle constructed previously into $2^{n-\tilde{k}}$ GU sets of $2^{\tilde{k}}$ points. This can be done by selecting one GU set of $2^{\tilde{k}}$ points and then select the other sets by taking the remaining points and associate a similar GU set from each one of them. This is possible since the entire n-dimensional hyperrectangle is GU. In order to select the first GU set of $2^{\tilde{k}}$ points, we can select a set of $\tilde{k}$ generating points, as for the construction of a linear $(n, \tilde{k})$ block code for which the $\tilde{k}$ independent n-dimensional rows of the generator matrix define the entire code. Geometrically, the construction can be done by executing the following algorithm.

- Step 1: $d_{min} = 0$. Select one point from the n-dimensional hyperrectangle as a reference. $i = 1$.

- Step 2: Select a new point from the n-dimensional hyperrectangle. $i = i + 1$. Consider this point as a generating point.

- Step 3: If $i \geq 3$, from this new point, select the $(i-2)$ points geometrically situated at the same distance than that between all previously constructed points (2 to $i-1$) and the first point. $i = i + (i-2)$.

- Step 4: If $i < 2^k$, go back to step 2.

- Step 5: Compute the minimum distance d'_{min} between all 2^k points. If $d'_{min} > d_{min}$, store this set of generating points, and $d_{min} = d'_{min}$.

- Step 6: Go back to step 2, until all possible generating points have been tried.

- Step 7: The best GU set of point is obtained by doing the (step 2, step 3) construction with the generating points stored in step 5.

Example 5.6.1 *We want to construct the best GU set of 4 points within the 3-dimensional hyperrectangle shown in Figure 5.17. We select point 1 as a reference at step 1. Suppose we select point 2 at step 2. Since $i = 2$ at step 3, we go back to step 2, and select for example point 3. Then, at step 3, the point situated from point 3 at the same distance than that between point 1 and 2 is 4. The minimum distance is then $d_{min} = \min\{1, r_2\}$. We store the generating points 2 and 3. After going through*

all possible generating points, we select 4 and 6 as generating points, which lead to the set of points with best minimum distance $\{1,4,6,7\}$. *Assuming* $r_2 < 1 < r_3$ *in this example, we have a GU set of 4 points with* $d_{min} = \sqrt{1+r_2^2}$.

Note that this algorithm is the geometric equivalent of an algebraic search for the best linear block code over the n-dimensional hypercube $\{0,1\}^n$. The $\tilde{k}$ rows of the generator matrix of a linear block codes correspond to the generating points of the algorithm. The step by step construction of the set of points is equivalent to avoiding generator matrices with linearly dependent rows. Note also that this algorithm can be directly implemented in the search of generating matrix for the trellis code defined in section 5.3.2, since the first index of each row of the trellis code generating matrix corresponds to the first DPB set.

Once the $2^{n-\tilde{k}}$ DPB sets have been selected, it is necessary to assign the $2^p - 1$ PB points associated to each DPB point. This can be done by constructing a sufficiently large section of a lattice constructed layers by layers as in the hyperrectangle construction. In theory, the entire lattice must be constructed, but since we only need to select points with lowest energy, it is only necessary to construct approximately 2^{p+n+1} points of the lattice. The n-dimensional lattice is defined by the n generating vectors of the lattice. By simplicity we limit our search to cubic lattices although as p and n increase, it becomes better to use lattices with better packing density. The size of the generating cube of the lattice must be larger than the minimum size among the n-dimensional DPB hyperrectangle sides. This introduces another parameter r indicating the ratio of the side of the lattice generating cube to the first side of the

n-dimensional DPB hyperrectangle. Therefore, we impose

$$\begin{cases} r > r_i \quad 2 \leq i \leq n \\ r > 1 \end{cases} \tag{5.6}$$

Once the points of the lattice with lowest energy have been assigned to each point of the DPB sets, we can start the generator search described in section 5.6.1. Note that the minimum distance between lattices associated to DPB points corresponds to the distance that we must use when computing the distance between these DPB points later on in the free distance computation. We now look at the case where $k \neq n$ and $\tilde{k} = n$.

Translation of the hyperrectangle

In the case where $\tilde{k} = n$, the entire hyperrectangle constitutes the first DPB set. We saw in Section 5.2.2 that is is necessary to have at least two distinct DPB sets on the first two states in order for the code to have sufficiently large distance or to avoid catastrophic labeling of the trellis. In this case, it is then necessary to translate the hyperrectangle. Yet, the two translated hyperrectangles already constitute a BEGU set. Therefore, the associated PB lattices must be constructed with the same basis than the DPB hyperrectangle as shown in Figure 5.18 (a) for the 2 dimensional case. Figure 5.18 (b) shows indeed an example where the PB lattices are square instead of rectangular as the DPB hyperrectangle. The PB lattice is highlighted for one of the DPB points. Hence, for this case, the resultant constellation is not GU. However, in Figure 5.18 (c), even though the PB lattices have the same shape than in Figure 5.18 (b), the translation of the second DPB is such that the constellation is GU.

Due to this extra constraint, in order for the PB lattices to be square, it is necessary that the DPB sets are hypercubes instead of hyperrectangles, or that the translation of the hyperrectangle leads to a GU constellation. This introduces a new n-dimensional parameter t corresponding to the vector of translation of the hyperrectangle. By simplicity, we consider, however, that the PB lattices are square, thus avoiding the non geometrically uniform case shown in Figure 5.18 (b).

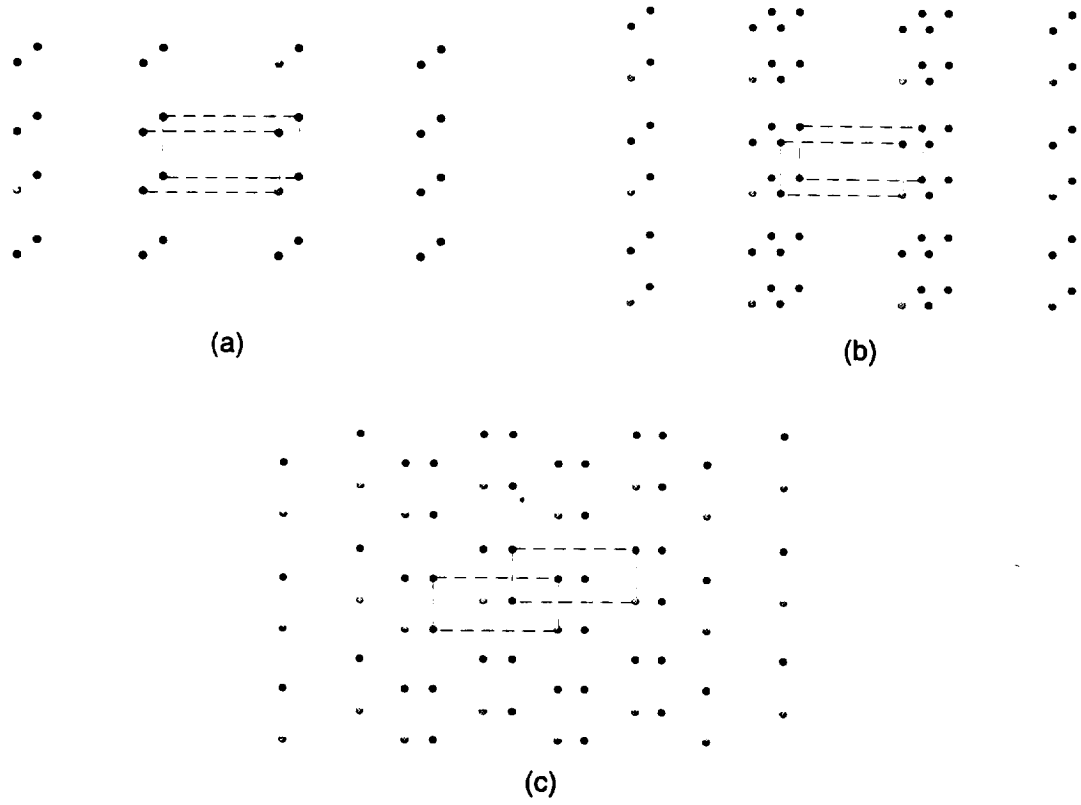

Figure 5.18: (a) Translated DPB sets with their associated rectangular PB sets (GU case), (b) Translated DPB sets with their associated square PB sets (non GU case), (c) Translated DPB sets with their associated square PB sets (GU case).

We now consider the case $k = n$ for which it is possible to situate all DPB sets and PB sets on an n-dimensional sphere (PSK) in order to use as little energy as

possible (see Section 5.3.1).

n-dimensional PSK constellations

For the case where $k = n$, we need to design a GU constellation on an n-dimensional sphere. For this purpose, we use the n-dimensional hyperrectangle and rotate it in order to obtain 2^{k+1} points on the sphere. This is to be sure that the partitioning of the 2^{k+1} points into two sets of 2^k points leads to two similar GU sets (n-dimensional hyperrectangle). Any matrix with determinant 1 is a rotation matrix. However, since we want to use a simulated annealing on the minimum number of parameters, we have to find the minimum number of rotation angles in n dimensions to rotate an object to any other possible position. The basic 2-dimensional rotation matrix R in a plan is given by

$$R = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}. \tag{5.7}$$

In 3 dimensions, there are three axis around which an object can be rotated. The three possible matrices are therefore given by

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x \\ 0 & -\sin\theta_x & \cos\theta_x \end{bmatrix} \quad R_y = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix}$$
$$R_z = \begin{bmatrix} \cos\theta_z & \sin\theta_z & 0 \\ -\sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.8}$$

A 3-dimensional object can then be rotated by successively applying each rotation matrix to each point of the object. More generally, in n dimensions there are $n(n-1)/2$ independent rotation matrices obtained by positioning $n-2$ one in all possible positions of an n-dimensional matrix and the basic 2 dimensional rotation matrix in the other 2 positions. This introduces $n(n-1)/2$ parameters θ_i for $1 \leq i \leq n(n-1)/2$.

However, although rotating a rectangle in two dimensions leads to a geometrically uniform constellations for any rotation angle, this is not the case for higher dimensional cases. In particular, for the 3 dimensional case, a cube has 8 vertices and only 6 faces. Therefore, it is impossible to rotate the cube in order to bring a vertex of the "new" cube at the center of a face of the "old" cube. It is possible to rotate the cube around one axis only, yielding a GU set as shown in Figure 5.19 and discovered previously be Slepian [50], but this does not set points uniformly on the sphere as it would be needed to get the largest possible distance between points.

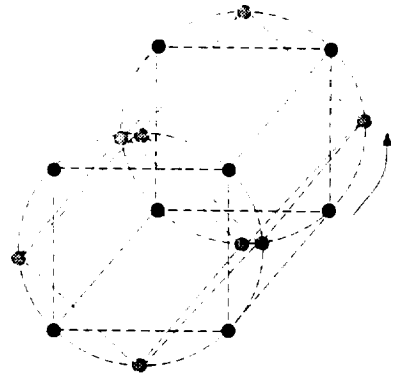

Figure 5.19: 3 dimensional cube rotated around one axis.

For the purpose of constructing codes with large distance and gain distance by constructing higher dimensional constellations, we allow all possible rotations but consider the minimum distance between the two rotated hyperrectangles as the min-

imum distance between any points of the first hyperrectangle and the rotated one. This still allows us to compute the free distance by only computing the Euclidean distance between the bottom path and the others, as for a geometrically uniform code. By letting the simulated annealing converge to the optimal parameters, we can hope that the resulting constellation is also geometrically uniform. We now describe the next step of the algorithm, common to all cases of k, p, and n, that is, the generator search.

Generator search

The generator search consists in finding the generating matrix described in Section 5.3.2 which yields the trellis code with the largest free distance. Once two DPB sets have been constructed by following the algorithm described in Section 5.6.1 if $\tilde{k} < n$, by translating the first hyperrectangle if $\tilde{k} = n$ as described in Section 5.6.1, or by rotating it if $k = n$ as described in Section 5.6.1, the branches leaving the two first states must be assigned with the DPB sets and their associated PB points. For the other states, a search through all possible DPB sets has to be done, which is similar to the search for convolutional code generators.

Note that for codes with rates lower than 1, it is possible to use the generator of optimal free distance convolutional codes, and try to modify the hypercube into a hyperrectangle by doing a simulated annealing on the rectangle side ratios r_i, for $2 \leq i \leq n$. The conversion from the usual convolutional code generator matrix notation to the notation described in Section 5.3.2 can be done by using the composite generator of convolutional codes as described in [9]. This is done by interleaving the

bits associated to each generator.

Example 5.6.2 *Let is convert the generator matrix of a 4 state, rate 2/3 convolutional code with squared free Euclidean distance 4 into the notation used in Section 5.3.2. The generator matrix is given by*

$$G = \begin{bmatrix} 11 & 01 & 11 \\ 01 & 10 & 10 \end{bmatrix}. \tag{5.9}$$

By taking the composite form of this matrix, we obtain

$$G_{comp} = \begin{bmatrix} 101 & 111 \\ 011 & 100 \end{bmatrix}, \tag{5.10}$$

which correspond to the points of the 3-dimensional cube

$$G_{GU} = \begin{bmatrix} 6 & 8 \\ 4 & 5 \end{bmatrix}, \tag{5.11}$$

However, note that the optimal free distance convolutional codes do not always lead to the best geometrically uniform codes. In particular, in Example 5.6.2, the generating matrix does not lead to an optimal free distance GU real number trellis code. There is indeed a code with squared free Euclidean distance of 5 and generating matrix

$$G_{GU} = \begin{bmatrix} 4 & 2 \\ 7 & 8 \end{bmatrix}. \tag{5.12}$$

Also, the best generator matrix can be the same for a given $\tilde{k}$ and n. Therefore, it is possible to obtain the same generator matrix for different k, which is similar

to Ungerboeck's results, where the same parity check matrix is used for different constellation expansions.

Simulated annealing on the constellation parameters

Constellation optimizations have been performed in the past [55, 56], but usually depending on only one parameter, in particular, the angle between the two DPB sets of two points in a 2-dimensional square, or between the two DPB sets on an 8PSK constellation. This corresponds to only one of the parameters that is adjustable on GU constellations as seen in previous sections, especially as the number of dimensions increases. In [56], the multidimensional constellations are indeed constructed with the Cartesian product of 2-dimensional constellations, adjustable with only one parameter.

However, the problem of optimization of a function depending on many parameters is not as simple as the optimization of a function of one parameter where simple analysis leads to the result. Therefore, one way of optimizing many parameters at the same time is to use a simulated annealing [26] on the parameters r_i, r, t and θ_i. Depending on k, p, and n, only certain parameters need to be adjusted as seen in the previous sections. In the next section, we indicate the results that we obtained from our implementation of the algorithm.

5.6.2 Results

The search for codes has been performed for various rates lower and greater than 1. Results are given in Tables 5.1, 5.2, 5.3, 5.6.2, 5.6.2, 5.6.2. Note that n-dimensional

codes are considered to be transmitted using an n-dimensional constellation with average energy 1. Therefore, the squared free Euclidean distance is normalized per dimension. In particular, to compare a rate 1/2 convolutional code free Hamming distance with a rate 1/2 GU real number trellis code squared free Euclidean distance, it is necessary to multiply the Hamming distance by 4 to obtain the squared free Euclidean distance of the code transmitted on a 1 dimensional BPSK, and divide it by $n = 2$ since it is transmitted with a 2-dimensional constellation. Therefore, the 2 state, rate 1/2 convolutional code with Hamming free distance 5, is equivalent to a rate 1/2 QPSK trellis code with free squared Euclidean distance $5 \times 4/2 = 10$.

Similarly, it is possible to compare a 4-dimensional trellis code obtained from the Cartesian product of two 2-dimensional constellations, such as Wei's codes [48] with an $n = 4$ GU trellis code by dividing the squared free Euclidean distance of the code constructed on the Cartesian product, since twice as much energy is needed to transmit 4 dimensions on two separate 2-dimensional constellations than on one 4-dimensional constellation. Therefore, the approach of this dissertation does not consider multi-dimensional trellis codes, but rather n-dimensional trellis codes, whatever the transmission scheme is. In particular, a 4-dimensional GU code can be transmitted using a multi 2D constellation, or a multi 1D constellation, as convolutional codes were first transmitted. When reading the table, however, it is necessary to multiply by the correct coefficient for comparison with existing codes.

The table only presents codes for which an improvement was noted as opposed to previous constructions. In particular for rates lower than 1, the codes are compared to optimal free distance binary convolutional codes [9]. For rates greater than 1 and

$n = 2$, the codes are compared to Ungerboeck's codes [3]. For n greater than 2, codes are compared to Wei's multi-dimensional codes [48]. The asymptotic gain in dB by using our codes instead of previously existing codes is given, as well as the asymptotic gain vs using an uncoded constellation of same dimension at the same transmission rate.

Depending on k, p, and n, only the concerned parameters described in the previous sections are given. In particular, for $\tilde{k} < n$ and $k \neq n$, r_i are specified for $2 \leq i \leq n$, d and r are specified if $p > 0$, for $\tilde{k} = n$, r_i, d, and t are specified. Finally for $k = n$, r_i and θ_i are specified. These parameters and the generator are sufficient to construct the code.

Finally, note that some GU trellis codes can be considered *asymptotically catastrophic* as some optimized parameters make the constellation merge into less points as noted in [55, 56]. This means that the free distance of the code tends to the value given in the table, as the parameter tends to their indicated value. However, if the parameters take exactly that value, the trellis code becomes catastrophic, which means that there is an infinite path which has lower Euclidean weight than the free distance. This path for a slightly different value of the parameters gains some weight as the number of symbols transmitted increases, thus avoiding the catastrophic property of the encoder, assuming the decoder waits enough time to decode.

Comments on the results

All codes with $k > 1$ have been constructed for trellises up to 16 states ($m = 4$). Only codes for which an improvement was noted when compared with existing codes are shown in the table. For codes with $k = 1$, codes have been constructed for trellises up to 2048 states ($m = 11$), but only codes with improvement over previously existing codes are shown in the table.

For rate 1 codes, where spherical constellations were used, note that the distance per dimension of the rate 2/2 code is 8 (resp. 9.172) (see [3]) for the 4 state (resp. 8 state) code, whereas the distance per dimension of the rate 3/3 code is 8 (resp. 8.37) (see Table 5.1 and 5.2). This shows that the distance per dimension decreases as n increases, which is contrary to what usually happens with block codes. Note also that for $n = 2$, no improvement was obtained by letting the rotation angle be different from $\pi/4$ (8PSK). In particular, we did not find any 16 state code with $d^2_{free} = 5.20$ as found in [55].

For rates greater than 1, improvement was noted for trellises with $T(1, p, m)$ topologies as noted in Section 5.3.1 by letting the square become a rectangle and by letting r be greater than $2\min\{1, r_i\}$, that is by having PB lattices larger than twice the DPB rectangle minimum side length. Also, as k increases, it becomes advantageous to select the points with lowest energy, as already mentioned by Wei [53].

For rates lower than 1, note that less improvement over binary codes is usually noted as n increases, since the number of points 2^n to select from becomes larger,

even for hypercubes (binary set). Note however that the distance per dimension can usually be made as large for $n = 2$ than for greater n by using real number codes instead of binary codes. For instance, the 4 state, rate 1/2 code only has distance 10 on the binary field, whereas is has distance 10.67 over the real field (as seen in Section 5.3.1), which is equal to the distance per dimension of the rate 1/3 binary convolutional code.

In average, the gain obtained by this new construction between .1 and .9 dB for non asymptotically catastrophic codes, and up to 1.25 dB for the asymptotically catastrophic 2 state rate 1/2 code. This gain is important especially for high rates codes, for which the coding gain versus using an uncoded constellation is not very high.

5.7 Conclusion

A new construction of geometrically uniform trellis codes based on geometric considerations has been presented. The constellation is geometrically constructed by following a decomposition of the trellis topology. This constructive process of the constellation allows us to identify generating branches on the trellis, which are sufficient to construct the entire trellis code. An algorithm was presented to optimize the free distance of a geometrically uniform trellis code on a given trellis topology. This algorithm also allows us to check the geometric uniformity of previously constructed codes such as some of Wei's multidimensional codes.

The construction of constellations using hyperrectangles instead of hypercubes was

proven to improve the free distance in many cases. This theory completely unifies the design of convolutional codes and trellis codes using geometric considerations, which can lead to the construction of codes for various rates and topologies.

Tables of codes

m	rate	d^2_{free} (real codes)	d^2_{free} (bin. codes)	d^2_{free} (trellis codes)	Gain *vs* uncoded (dB)	Gain *vs* coded sym.
2	5/2	.392	-	5/6 (64QAM) (.381)	3.13	.123
3	5/2	.491	-	5/6 (64QAM) (.476)	4.11	.135
4	5/2	.595	-	5/6 (64QAM) (.571)	4.94	.178
2	4/2	.807	-	4/5 (32AMPM) (.762)	3.05	.249
3	4/2	1.0	-	4/5 (32AMPM) (.952)	3.97	.214
4	4/2	1.21	-	4/5 (32AMPM) (1.14)	4.80	.259
2	3/2	1.68	-	3/4 (16QAM) (1.6)	4.57	.212
1	1/2	$8^\dagger$	6	-	3.01	1.25
2	1/2	10.67	10	-	4.26	.281
3	1/2	13.33	12	-	5.23	.456
7	1/2	21.33	20	-	7.27	.280
9	1/2	25.14	24	-	7.98	.202

Table 5.1: Two-dimensional real number trellis codes

m	rate	d^2_{free} (real codes)	d^2_{free} (bin. codes)	d^2_{free} (trellis codes)	Gain *vs* uncoded (dB)	Gain *vs* coded sym.
2	4/3	1.17	-	-	3.11	
3	4/3	1.77	-	-	4.91	
2	3/3	2.67	-	-	3.01	
3	3/3	2.79	-	-	3.21	
2	2/3	5.00	4	-	2.73	.969
3	2/3	6.40	5.33	-	3.80	.794
1	1/3	$8^\dagger$	6.67	-	3.01	.789
5	1/3	18	17.33	-	6.53	.165
7	1/3	22.61	21.33	-	7.52	.253
9	1/3	27.29	26.67	-	8.33	.100

Table 5.2: Three-dimensional real number trellis codes

m	rate	d^2_{free} (real codes)	d^2_{free} (bin. codes)	d^2_{free} (trellis codes)	Gain *vs* uncoded (dB)	Gain *vs* coded sym.
2	5/4	.875	-	-	2.43	-
3	5/4	1.23	-	-	3.91	-
4	5/4	1.53	-	-	4.86	-
2	3/4	3.2	3	-	2.04	.280
2	2/4	5.33	5	-	4.26	.281
3	2/4	6.4	6	-	5.05	.280
2	1/4	10.67	10	-	4.26	.281
3	1/4	13.33	13	-	5.23	.109
7	1/4	22.85	22	-	7.56	.165

Table 5.3: Four-dimensional real number trellis codes

Tables of codes parameters

m	rate	p	Generator	d	r	r_i	t	θ_i (rad)
2	5/2	4	$(4,2,4)$	-	2.0	1.288	-	-
3	5/2	3	$\begin{pmatrix} 4,6,2 \\ 3,4 \end{pmatrix}$	$(-3.32\,10^{-2},$ $-7.04\,10^{-2})$	1.141	1.	$(.213,$ $-.654)$	-
4	5/2	3	$\begin{pmatrix} 4,6,3 \\ 3,8,4 \end{pmatrix}$	$(-.311,$ $-6.4\,10^{-2})$	2.415	.979	$(-.130,$ $-.432)$	-
2	4/2	3	$(4,3,4)$	-	2.375	1.134	-	-
3	4/2	2	$\begin{pmatrix} 4,6,3 \\ 3,4 \end{pmatrix}$	$(.111,$ $.111)$	1.125	1.	$(.222,$ $-.222)$	-
4	4/2	2	$\begin{pmatrix} 4,6,3 \\ 3,8,4 \end{pmatrix}$	$(9.66\,10^{-2},$ $-1.08\,10^{-3})$	1.5	.959	(.32, 0.)	-
2	3/2	2	$(4,3,4)$	-	1.547	.774	-	-
1	1/2	0	$(4,2)$	-	-	0.	-	-
2	1/2	0	$(4,2,4)$	-	-	.707	-	-
3	1/2	0	$(4,4,2,4)$	-	-	.707	-	-
7	1/2	0	$(4,3,3,2,$ $2,4,2,4)$	-	-	1.414	-	-
9	1/2	0	$(2,3,1,2,2,$ $4,1,2,4,2)$	-	-	1.155	-	-

Table 5.4: Two-dimensional constellation parameters

m	rate	p	Generator	d	r	r_i	t	θ_i (rad)
2	4/3	3	$(6,7,8)$	-	1.41	(.833,.998)	-	-
3	4/3	2	$\begin{pmatrix} 6,8,6 \\ 7,7 \end{pmatrix}$	-	2.36	(1.184,1.)	-	-
2	3/3	2	$(2,4,2)$	-	-	(1.,1.)	-	(0.,.337,.420)
3	3/3	1	$\begin{pmatrix} 4,8,3 \\ 2,4 \end{pmatrix}$	-	-	(.954,1.099)	-	(.808,0.,1.57)
2	2/3	0	$\begin{pmatrix} 4,2 \\ 7,8 \end{pmatrix}$	-	-	(.707,1.)	-	-
3	2/3	0	$\begin{pmatrix} 7,5,4 \\ 8,6 \end{pmatrix}$	-	-	(1.,.707)	-	-
1	1/3	0	$(6,8)$	-	-	(0.,1.)	-	-
5	1/3	0	$(8,2,4,6,7,8)$	-	-	(.707,1.)	-	-
7	1/3	0	$(8,4,2,8,$ $3,6,2,8)$	-	-	(.656,1.222)	-	-
9	1/3	0	$(8,2,4,7,3,$ $4,6,7,6,8)$	-	-	(1.154,.613)	-	-

Table 5.5: Three-dimensional constellation parameters

C-3

m	rate	p	Generator	d	r	r_i	t	θ_i (rad)
2	5/4	4	$(\ 15,16,16\ )$	-	1.72	(.804,1.50,1.18)	-	-
3	5/4	3	$\begin{pmatrix} 12,11,13 \\ 13,16 \end{pmatrix}$	-	1.03	(.51,1.,1.)	-	-
4	5/4	2	$\begin{pmatrix} 3,9,13 \\ 12,10 \\ 6,16 \end{pmatrix}$	-	2.05	(1.04,1.05,.95)	-	-
2	3/4	0	$\begin{pmatrix} 10,14,11 \\ 15,16 \end{pmatrix}$	-	-	(1.,.817,1.)	-	-
2	2/4	0	$\begin{pmatrix} 10,14 \\ 15,15 \end{pmatrix}$	-	-	(.707,1.414,.707)	-	-
3	2/4	0	$\begin{pmatrix} 15,11,13 \\ 4,16 \end{pmatrix}$	-	-	(1.05,.850,.876)	-	-
2	1/4	0	$(\ 16,15,16\ )$	-	-	(.933,1.115,.204)	-	-
3	1/4	0	$(\ 16,2,16,16\ )$	-	-	(.381,1.588,2.019)	-	-
7	1/4	0	$(16,4,6,13,$ $16,14,4,16)$	-	-	(1.414,1.224,.577)	-	-

Table 5.6: Four-dimensional constellation parameters

CHAPTER 6

CONCLUSION

This dissertation has shown that the construction of codes over the real numbers can be advantageous over binary numbers. First, block codes were presented as sets of codewords in finite dimensional space and trellis codes as set of sequences in infinite dimensional space. Although block codes in the binary field can be constructed and decoded using algebraic techniques, convolutional codes benefit from the binary field by imposing linearity between the set of code sequences, which simplifies the construction and the analysis of the code performance. For convolutional codes, however, algebraic techniques have not so far lead to the construction of good codes, and the best decoding techniques involve algorithms which do not make use of the binary algebraic structure.

In Chapter 2, we showed that it is possible to construct good binary convolutional codes with a new technique based more on the statistical properties of the code generators, instead of its algebraic properties. This led to the construction of new codes with much larger constraint length than previously constructed codes. It was

shown that good trellis codes usually exhibit a constant row distance equal to the minimum free distance of the code. This important concept is even more useful in the construction of real number trellis codes. In Chapter 3, we proved that the bounds on the minimum free distance of codes over the real field are larger than the bounds on the minimum free distance of binary codes of the same rate, especially for rates greater than 1. In addition, the decoding algorithms available for trellis codes are not restricted to binary numbers, because they can perform soft decision decoding which make use of the real output from the channel. Therefore, it seemed advantageous to construct trellis codes over the real field as opposed to the binary field.

However, several difficulties are encountered when constructing trellis codes over the real field. One such difficulty is the lack of linearity between code sequences, which make the computation of the minimum free distance difficult. Another difficulty is that it is not possible to do a combinatorial search to find the best code generators. This last difficulty was avoided in Chapter 4 by developing a search algorithm adapted to the density of the real field, that is, a simulated annealing on the code parameters [26]. The simulated annealing allowed us to construct some codes over the real field with a better minimum free distance than binary convolutional codes with the same trellis topology. However, the searches had to be executed over a larger number of parameters than binary convolutional codes, and the computation of the minimum free distance was much more difficult due to the lack of linearity. In Chapter 5, we used the concept of geometric uniformity to extend the concept of linearity in the binary field to the real field. The free distance can be computed from one sequence only, and the number of parameters to select can be reduced to the same number of

parameters than for a binary convolutional code. This allowed us to construct real number trellis codes of all rates and relatively large constraint length with optimized free distance.

In the next section, we present the concepts seen in this dissertation in a more geometric way, which explains the construction of real and binary block and trellis codes in terms of sphere packing arguments.

6.1 Unification of the theory for constructing binary and real number codes

As it was noted in Chapter 3, constructing codes is equivalent to finding the best sphere packing in a n-dimensional space. If we are constructing a block code, the dimension n is fixed. For a trellis code, the number of dimensions separating two code sequences is variable. However, some concepts can be studied and compared for both types of codes.

6.1.1 Low rate codes

When the number of codewords constructed in n dimensions is less than 2^n, then the rate is lower than 1, and it is possible to obtain the best minimum distance by placing all codewords on a sphere of energy 1. In particular, binary code words or sequences are constructed on the n dimensional hypercube contained in this sphere. Although the best real number block code would have all codewords positioned on the n dimensional sphere with equal distances between them (2^k polytope), good real number block codes result from taking the best 2^k points of the binary n-dimensional

hypercube. Moreover, imposing linearity between the binary codewords helps in constructing and studying the code performance without limiting the distance achievable for a block code of that rate.

Some gain can be obtained by using the best 2^k dimensional polytope as opposed to the best set of 2^k points in the n-dimensional hypercube. Thus, some gain is expected by using real number codes as opposed to binary codes. An approach which maintains linearity or group structure among code words, while slightly improving the distance involves degenerating the n-dimensional hypercube into a hyperrectangle, which is a geometrically uniform structure. Figure 6.1 shows the best polytope with 3 codewords on a 2-dimensional sphere, and the best set of 3 codewords taken from the 2-dimensional square. Note the difference between the minimum distances. However, for 2 or 4 codewords, the best polytope is a hypercube and it does not change the minimum distance to restrict our codewords to the vertices of the square.

For convolutional codes, the problem is slightly more difficult to perceive geometrically, since code sequences are not all constructed over the same number of dimensions. However, given a length l, it is possible to look only at all remerged paths of that length through the trellis. The set of such paths constitutes a block code with 2^{kl-m} codewords in an nl dimensional space. Yet, since the constituent code words are not all separated by the same number of dimensions, they must be positioned optimally within their sub-dimensional space, which can generally not produce a rate $(kl-m)/nl$ block code as good as the best block code of the same rate. However, using real numbers allows us to construct codewords in sub-dimensional hyperrectangles instead of hypercubes, and allows us to reach the optimal free distance

given by the tightest upper bounds on the minimum distance of block codes.

Recall the case of the 4 state rate 1/2 convolutional code described in Chapters 4 and 5, for which the $n \times 4 = 8$ dimensional code sequence is separated by a distance 12 from the all zero code sequence while the $n \times 3 = 6$ dimensional code sequence is separated by a distance 10 from the all zero code sequence. By constructing the code sequences over a hyperrectangle instead of a hypercube, both distances become equal, and the code has a better minimum free distance.

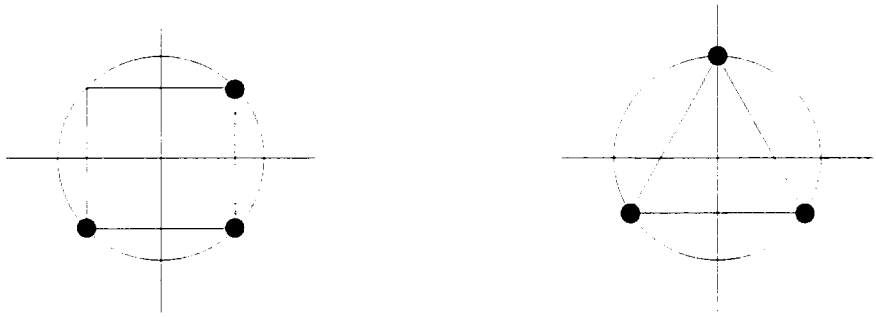

Figure 6.1: 2-dimensional hypercube and polytope with 3 codewords.

6.1.2 High rate codes

When the number of codewords constructed in n dimensions is greater than 2^n, then the rate is greater than 1, and the best polytope with 2^k points on the same n dimensional sphere does not optimize the minimum distance between points, which is the case for low rate codes. As an example, compare the minimum distance between 8 codewords constructed on a circle and the optimal positioning in Figure 6.2. Therefore, it becomes important to achieve the best packing in n dimensions and take the set of codewords from this packing with lowest energy. Note that codewords do not

all have the same energy as it was the case for codes with rate lower than 1.

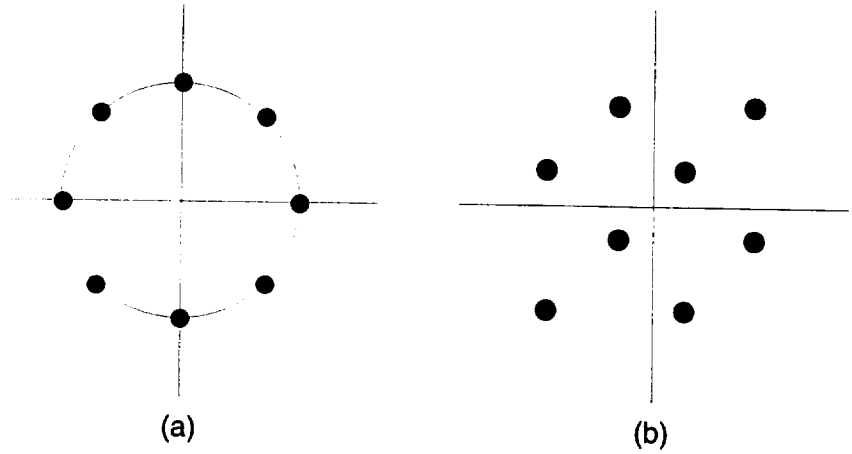

Figure 6.2: Construction of 8 codewords on (a) a 2-dimensional sphere and (b) optimal positions.

One way of constructing real number codewords for an n dimensional block code is to use the best lattice in n dimensions and take the points with lowest energy. However, as n increases, our knowledge of the best lattice packings decreases, and it is necessary to find a better way of constructing codewords. One possibility is to simply translate in all directions the best low rate block code with 2^k codewords constructed on an n-dimensional hypercube, which creates a geometrically uniform structure. In other words, each codeword is translated in a lattice pattern in N dimensions (dimensionality of the constellation generally lower than n). We must check that the minimum distance between codewords of two translated versions is greater than the minimum distance between codewords within the same hypercube. Figure 6.3 shows a construction of 16 codewords in a two dimensional space by translating a structure with (a) 2 codewords per square, (b) 3 codewords per square, and (c) 4 codewords per square. Note that the sets of points are equivalent in (a) and (c), and optimize

the ratio minimum distance to energy. However, (b) does not optimize the distance. Varying $\tilde{k}$ allows us to optimize the decomposition. In this example, $n = N$, but the best way of increasing the overall minimum distance of the code is to let n much greater than N in order to construct a low rate block codes with large distance. The distance between hypercubes in N dimensions should then be comparable to the distance of the codewords in n dimensions within the hypercube, which suggests that the size of the hypercubes should become smaller as n becomes larger than N.

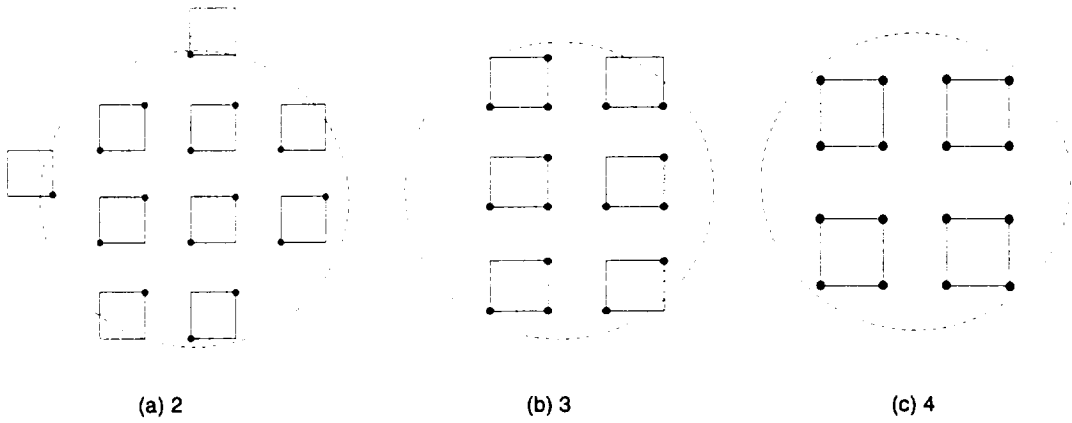

Figure 6.3: Construction of high rate codes by assembling low rate codes.

Looking at the previous construction when using Ungerboeck's concept of set partitioning [3], the whole constellation is simply divided into $2^{\tilde{k}}$ subsets of translated points. The minimum distance of the code is then simply given by the minimum between the translation distance and the codewords distance. In other words, each of the translated hypercubic structure is coded by selecting the best set of $2^{\tilde{k}}$ points. If we do not use coding, we select all points of each hypercube (rate 1 code), thus selecting the lowest energy points of the Z^n lattice, which is not the best packing for $n > 1$. This new way of looking at the constellation provides a better idea of how to

optimize the distance. It is indeed better to construct an optimal low rate code on an n-dimensional hypercube, and then translate the hypercube with the best pattern, than first constructing the entire constellation and try to decompose it afterwards.

When using trellis codes instead of block codes, the idea is similar. A low rate $\tilde{k}/n$ trellis code creates a set of code sequences within each translated hypercube, thus increasing the distance between the coded points in the hypercube. The minimum distance of the code is then given by the minimum between the translation distance (parallel branches) and the minimum free distance of the trellis code. For trellis codes, we usually use $n = N$ as seen in Chapter 4, since the trellis structure develops sequences in an infinite dimensional space anyway. Here again, the difference with block codes is that the code sequences constructed on the main hypercube are of various dimensionalities, whereas the translated points are all in the n dimensional space. Note that varying $\tilde{k}$ here is similar to varying $\tilde{k}$ in the block code case, since it provides a way of adjusting the number of hypercubes to use and the number of encoded points within each hypercube.

The purpose of using real number codes as opposed to the regular Z^n lattice, is to be able to adjust the length of the hyperrectangle sides, in order to equate the distances between sequences of different dimensionalities. Also, we can adjust the translation vector such that the minimum distance between translated points and the minimum free distance of the trellis code are equal.

Note that a new problem arises with high rate codes: each hypercube does not have the same energy. It is obviously not optimal to simply choose which hypercube is transmitted by using the remaining $k - \tilde{k}$ information bits (uncoded). This leads

to the concept of *shaping* which maps the remaining uncoded information bits to the translated hypercubes in such a way that the overall energy used when transmitting the code is optimal. Shaping was first introduced by Calderbank and Ozarow [60], after Forney and Wei [61] had proved that it is optimal to select signal points with a Gaussian probability distribution centered at the origin of the constellation. Later, Forney [62] developed an implementation scheme using a block or convolutional code. The next section will describe the main idea of shaping, and we will show that our construction of real number trellis codes can be improved by shaping as much as regular trellis codes constructions are.

6.2 Shaping on high rate codes

The idea of shaping is to optimize the shape of the constellation in order to optimize the average amount of energy needed to transmit symbols. It is based on the simple concept that symbols should be distributed on a sphere instead of a cube, since symbols close to the vertices of a cube have greater energy. The gain of energy by distributing symbols on a sphere as opposed to a cube increases with the dimensionality of the space. In particular, the gain of an n-dimensional sphere to an n-dimensional cube goes to $\pi e/6$ which corresponds to an asymptotic gain of 1.53 dB. However, since we are transmitting symbols in constellations of finite dimensionality, it is necessary to create a spherical constellation over a multiple use of the constellation. Suppose for example that we are working with a 1-dimensional constellation, and we send two successive symbols, which can therefore be represented in a 2-dimensional constella-

tion obtained by the Cartesian product of the 1-dimensional constellation with itself. Assuming we can send all symbols in the 1-dimensional constellation, the Cartesian product is a square as shown in Figure 6.4. Assuming we only want to use the symbols within the 2-dimensional circle, some pairs of symbols cannot be successively sent, specifically $(a,a),(a,d),(d,a),(d,d)$. If we look at the probability distribution of sending each symbol, b and c are sent twice as many times as a and d. We would be able to send twelve different symbols, by sending a, b, c, and d with unequal probability. If we were extending this concept to an infinite dimensional spherical constellation, the symbols on the 1-dimensional constellations should be sent with a Gaussian distribution P_d [61]. Therefore, reciprocally, if we transmit symbols of the constellation with a Gaussian distribution probability, we obtain a spherical packing after an infinite number of constellation uses.

Different schemes have been studied to send symbols with a Gaussian-type probability distribution. The first type of scheme was introduced by Calderbank and Ozarow [60] by following the same type of idea than Ungerboeck's for the coding part. It consists in using the remaining $k-\tilde{k}$ bits in input of a block code with some redundancy. Codewords are then mapped onto the constellation such that more codewords are assigned to symbols with low energy and less codewords are assigned to symbols with high energy. Thus, the mapper is obtained by partitioning the constellation into energy regions. By using 12-dimensional block codes and 4 different regions, a shaping gain of about .9 dB can be obtained. Another scheme introduced by Forney [62] uses a convolutional code for the shaping code. The block diagram of the whole coding scheme combining the coding and the shaping codes is shown in

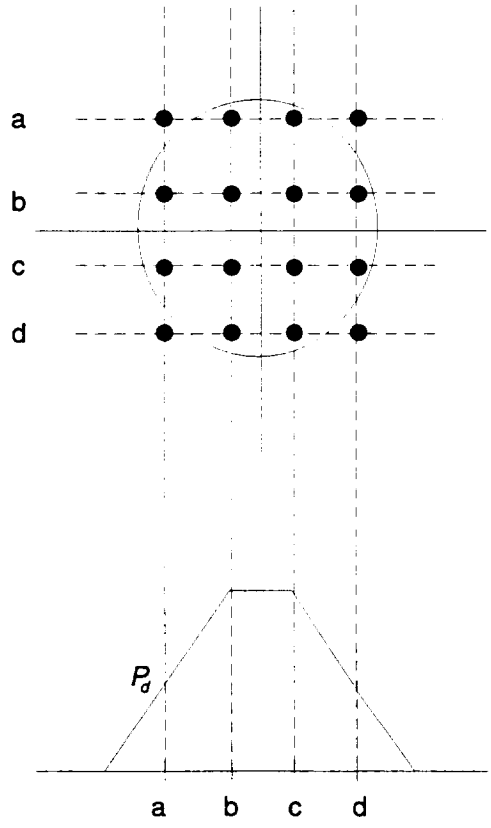

Figure 6.4: Shaping on the Cartesian product of 1-dimensional constellations.

Figure 6.5.

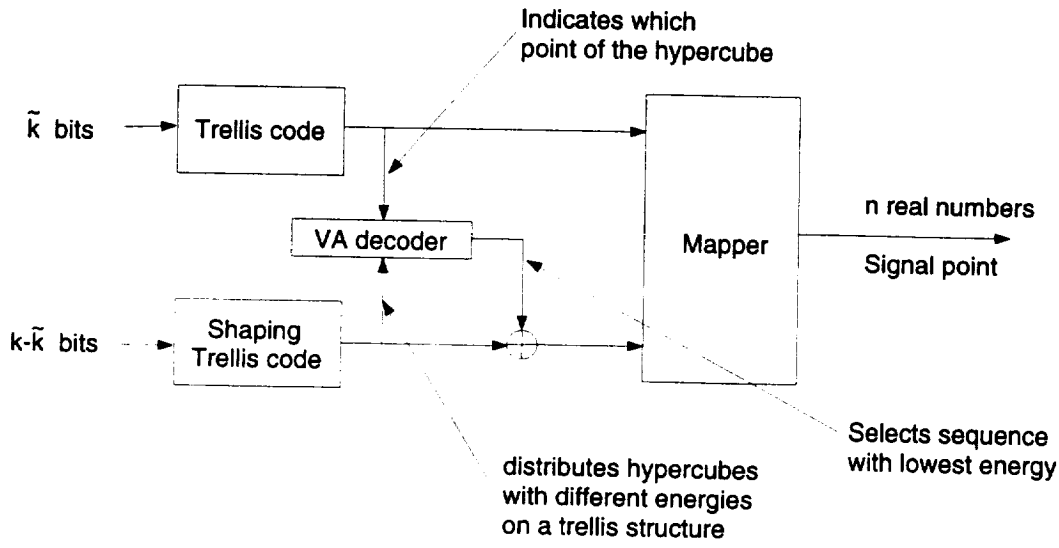

Figure 6.5: Trellis shaping block diagram.

Similar to the coding problem, the total number of points in the constellation needs to be expanded in order to use the shaping trellis code. That is, hypercubes are added to the original coding scheme. The shaping trellis code then creates a set of sequences representing which hypercube is used in the transmission on the different branches of the shaping trellis structure. A Viterbi decoder selects over an infinite number of time units the sequence with lowest energy. This creates the desired shaping.

The important point of shaping is that the selection of the hypercube transmitted is an independent process from the selection of which point is selected within the hypercube, as long as the noise is not too large, which would confuse the selection of points within the hypercube and between hypercubes, thus destroying the advantage of shaping. Therefore, for large SNR, the optimization of the free distance between

points within the hypercube is independent from the shaping process. In particular, in this thesis, it was proven that equating distance between sequences optimizes the free distance of the code. Moreover, for codes with rates greater than 1, we observed that the distance between hypercubes determined by the minimum distance between points on the parallel branches of the trellis should also be equal to the free distance between coded branches. This remains valid when adding shaping. However, the addition of shaping requires a constellation expansion, which indicates a modification of the constellation constructed for the trellis code.

Therefore, the optimization of the constellation for the trellis code combined with shaping should be studied in future research. In particular, if we suppose that we use a rate k_s/n_s shaping code, should we design the constellation for the real number trellis code with $k+n_s-k_s$ input bits, such that the expanded constellation is optimized, or rather select the constellation completely independently from the shaping problem ? Also, can the constellation be optimized for both shaping and trellis codes at the same time ? These questions need to be answered as we want to reach channel capacity with our coding scheme, especially since between .3 and .5 dB improvement could be obtained by optimizing the constellation, while theoretically up to 1.5 dB can be obtained with shaping.

Since this dissertation investigated a geometric approach for real number coding, which brought the same advantages than the usual algebraic approach for binary codes, that is, linearity, and generating branches in the trellis, it is interesting to explain the connection between these two approaches. The next section studies the equivalence between the algebraic description in the binary field and the geometric

description in the Euclidean space. This will lead to further recommendations for future work.

6.3 Algebraic techniques for geometric constructions

Binary block codes were first discovered for correcting errors by adding redundancy to information bits. Implementation purposes required to work within the binary field, and most of the construction and decoding techniques were discovered using the binary field. Research on the sphere packing problem brought a new approach to error control coding, that is, positioning points in the Euclidean space instead of the binary field. This more geometrical concept lead to the description of a block code as the best set of points taken from an n-dimensional hypercube. With the idea of combining coding in the binary field with new types of constellations, high rate codes were then introduced by Ungerboeck [3]. However, Ungerboeck's construction idea of set partitioning the constellation to assign binary digits to points in the space was done without geometrical motivation. The new concept of geometric uniformity, similar to the linearity within the binary field was then discovered for the codes seen in the Euclidean space. Recently, researchers have tried to find an algebraic connection between an arbitrary binary labeling by set partitioning of the points in the Euclidean space, a binary code construction, and the resulting geometric uniformity of the code in the Euclidean space.

In this section, we use the geometric description of the last section to give a constructive algebraic support to codes in the Euclidean space. It was explained

that low rate block and convolutional codes are constructed by selecting points from an n-dimensional hypercube. Since a hypercube is described by two positions per dimension, it is natural to assign a 0 to the first vertex in each direction and 1 to the other vertex. Figure 6.6 shows this obvious labeling for a 3-dimensional cube. Similarly, high rate codes were described in the previous section as a set of translated hypercubes in which a low rate code is constructed. Therefore, additional binary labeling can simply be given to each hypercube to designate the hypercube. This constructive approach of a constellation gives more insight to the problem of assigning a binary label to a point of the constellation, than the set partitioning technique does.

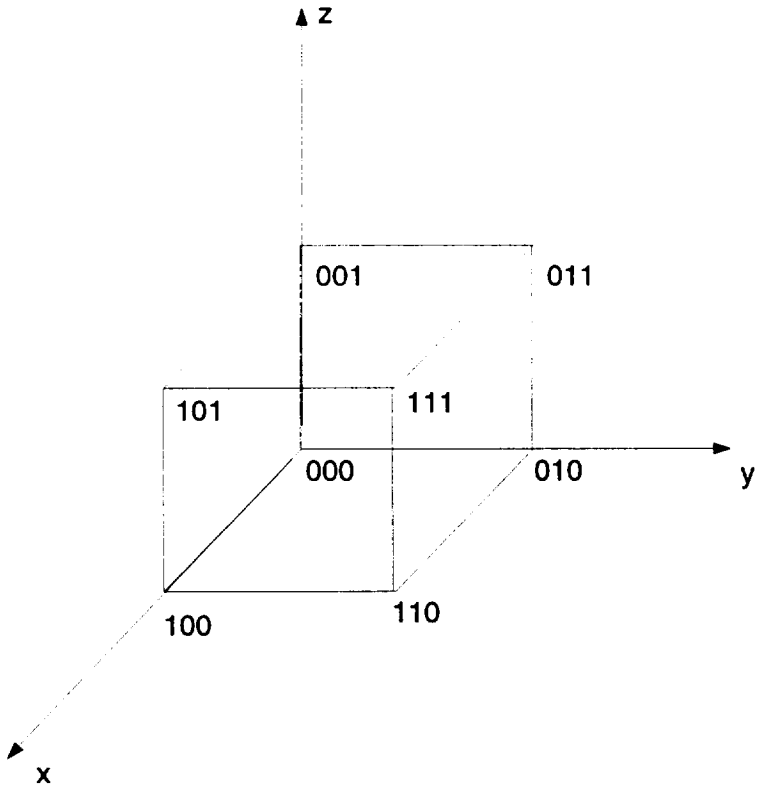

Figure 6.6: Binary labeling of a 3-dimensional cube.

The interesting aspect of this natural labeling procedure is that a geometric sim-

ilarity can be expressed by a binary difference. Specifically, when one set of points is defined, geometrically similar cosets can be constructed by selecting a point and adding the same binary labels. For instance, if the square (000,100,110,010) is taken as reference in the cube in Figure 6.6, adding labels (100,110,010) to another point defines a geometrically similar set. For example, from point 001, adding the labels of the first square yields (001,101,111,011), which is the translate of the first square, thus geometrically similar. This explains why all linear codes over the binary field are also geometrically uniform.

The question remains whether all geometrical similar cosets can be obtained by this binary operation. The answer is certainly no, since for instance, (001,101,111,011) is geometrically similar to (101,111,011,001) while applying the binary construction from the point 101 leads to (101,001,011,111). This explains Wei's geometrically uniform trellis code constructed over a nonlinear underlying convolutional code described in Chapter 5. Therefore, a more interesting question is whether all geometrical uniform cosets can be constructed algebraically from a given set, given a particular binary labeling, or whether the binary field does not lead to all possible representation of geometrically uniform codes.

This question is being studied by Forney and Trott [63] who defined a new class of codes, called group codes, described over different algebraic groups. In particular, the description of geometric generating matrix of a trellis code in section 5.3.2 does not simply lead to all possible cosets on the branches of the trellis, and no parity-check matrix or syndrome generating matrix can be systematically constructed with this approach. Future research should therefore pursue the geometric construction seen in

this dissertation, since it allows a better understanding of the adjustable parameters to optimize the code distance, but find new techniques to adapt an algebraic structure to the constellation. These techniques should probably be oriented towards a constructive approach of the constellation instead of arbitrary set partitioning techniques.

6.4 Summary

In this dissertation, the construction of real number trellis codes has been viewed from a geometrical point of view. We observed that the topology of the structure leads to a systematic construction of some parts of the trellis such as the construction of diverging branches sets and parallel branches sets, as well as the optimization of the constellation parameters. Yet, the remerging aspect of the trellis still makes it difficult to systematically construct the generating branches in order to optimize the resulting minimum free distance of the code. Therefore, a search through all possible generators is still the best way to construct trellis codes. However, we presented in Chapter 2 a search technique based on the statistical properties of the generator for binary rate $1/n$ convolutional codes, which accelerates the process of finding good generators. In Chapter 3, the sphere packing problem was introduced to give a more geometrical approach to the problem of coding theory, and bounds showed the possible improvement by using the entire Euclidean space to construct codes as opposed to using the Z^N binary lattice. In Chapter 4, the construction technique of trellis codes by decomposition into a convolutional code and a mapper was presented.

Then, direct optimization of trellis codes was proposed. The direct construction method led to some trellis codes with better free distance than previously constructed codes. Finally, in Chapter 5, the concept of geometric uniformity was developed to simplify the construction of real number trellis codes with optimized free distance. This led to a new description of trellis codes where the constellation is adapted to the topology of the trellis code structure. Some questions remain for future research in the area. In particular, future research may investigate the addition of shaping to the construction of optimal trellis codes, and develop a more general algebraic description of geometrically uniform real number trellis code.

BIBLIOGRAPHY

[1] H. Nyquist, "Certain topics in telegraph transmission theory," *AIEE Trans.*, vol. 47, pp. 617–644, 1928.

[2] C. E. Shannon, "Communication in the presence of noise," in *Proc. IRE*, vol. 37, pp. 10–21, January 1949.

[3] G. Ungerboeck, "Channel coding with multilevel/phase signals," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 55–67, Jan. 1982.

[4] A. Papoulis, *Probability, Random Variables, and Stochastic Processes.* McGraw-Hill Publishing Company, 1984.

[5] D. Slepian, "Bounds on communication," in *Bell system Technical Journal*, vol. 42, pp. 681–707, 1963.

[6] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Inf. Control*, pp. 68–79, March 1960.

[7] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math.*, pp. 300–304, June 1960.

[8] D. E. Muller, "Applications of boolean algebra to switching circuit design and to error detection," *IRE Trans.*, pp. 6–12, September 1954.

[9] S. Lin and D. J. Costello, Jr., *Error Control Coding.* Prentice-Hall, Englewood Cliffs NJ, 1983.

[10] J. G. Proakis, *Digital Communications.* McGraw-Hill Publishing Company, 1989.

[11] J. M. Wozencraft and I. M. Jacobs, *Principles of Communication Engineering.* Waveland Press, Inc., 1965.

[12] P. Elias, "Coding for noisy channels," *IRE Conv. Rec.*, pp. 37–47, 1955.

[13] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 260–269, April 1967.

[14] J. L. Massey and T. Mittelholzer, "Convolutional codes over rings," in *Proceedings 4th Joint Swedish-Soviet Int. Workshop on Inform. Theory*, pp. 14–18, August-Sept 1989.

[15] G. D. Forney, "Geometrically uniform codes," *IEEE Trans. Inform. Theory*, vol. IT-37, pp. 1241–1260, Sept. 1991.

[16] H. A. Loeliger, "Signal sets matched to groups," *IEEE Trans. Inform. Theory*, vol. IT-37, pp. 1675–1682, November 1991.

[17] O. Ytrehus, "Binary convolutional codes of high rate." Nov. 1991.

[18] E. Prange, "Cyclic error-correcting codes in two symbols," *Air Force Cambridge Research Center, Cambridge, Mass.*, vol. AFCRC-TN-57, 103, September 1957.

[19] J. Justesen, "New convolutional code constructions and a class of asymptotically good time-varying codes," *IEEE Trans. Inf. Theory*, vol. IT-19, pp. 220–225, March 1973.

[20] J. L. Massey, D. J. Costello, Jr., and J. Justesen, "Polynomial weights and codes constructions," *IEEE Trans. Inf. Theory*, vol. IT-19, pp. 101–110, January 1973.

[21] Y. Levy and D. J. Costello, Jr., "Constructing convolutional codes from quasi-cyclic codes," in *IEEE Workshop on Coding and Quantization*, October 1992.

[22] R. M. Tanner, "Convolutional codes from quasi-cyclic codes: A link between the theories of block and convolutional codes," in *Computer Research Laboratory, Technical Report, USC-CRL-87-21*, November 1987.

[23] J. L. Massey and M. K. Sain, "Inverses of linear sequential circuits," *IEEE Trans. Comput.*, vol. C-17, pp. 330–337, April 1968.

[24] S. W. Golomb, *Shift-register sequences.* Aegean Park Press, 1982.

[25] Y. Levy and D. J. Costello, Jr., "A deterministic approach to finding good convolutional encoders," in *Sesquicentennial Graduate Symposium, Univ. of Notre Dame*, pp. 42–43, February 1992.

[26] A. A. E. Gamal, L. A. Hemachandra, I. Shperling, and V. K. Wei, "Using simulated annealing to design good codes," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 116–123, Jan. 1987.

[27] C. A. Rogers, *Packing and Covering.* Camb. Univ. Press, 1964.

[28] J. Leech and N. J. A. Sloane, "Sphere packings and error-correcting codes," *Can. J. Math.*, vol. XXIII, no. 4, pp. 718–745, 1971.

[29] G. A. Kabatianski and V. I. Levenshtein, "Bounds for packings on a sphere and in space," *Problemy Peredachi Informatsii*, vol. 14, pp. 3–25, 1978.

[30] E. N. Gilbert, "A comparison of signalling alphabets," *Bell Syst. Tech. J.*, vol. 31, pp. 504–522, 1952.

[31] R. J. McEliece, E. R. Rodemich, H. C. Rumsey, Jr., and L. R. Welch, "New upper bounds on the rate of a code via the delsarte-macwilliams inequalities," *IEEE Trans. Inform. Theory*, vol. IT-23, pp. 157–166, 1977.

[32] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes.* North-Holland Mathematical Library, 1988.

[33] J. H. Griesmer, "A bound for error-correcting codes," *IBM J. Res. Develop.*, vol. 4, pp. 532–542, 1960.

[34] A. R. Calderbank, J. E. Mazo, and V. K. Wei, "Asymptotic upper bounds on the minimum distance of trellis codes," *IEEE Trans. on Communications*, vol. COM-33, pp. 305–309, April 1985.

[35] J. A. Heller, "Short constraint length convolutional codes," in *Jet Propulsion Lab., California Inst. Technol., Space Programs Summary 37-54*, vol. 3, pp. 171–177, 1968.

[36] J. Layland and R. McEliece, "An upper bound on the free distance of a tree code," in *Jet Propulsion Lab., California Inst. Technol., Space Programs Summary, 37-62*, vol. 3, pp. 63–64, 1970.

[37] G. J. Pottie and D. P. Taylor, "Sphere-packing upper bounds on the free distance of trellis codes," *IEEE Trans. Inform. Theory*, vol. IT-34, pp. 435–447, May 1988.

[38] A. R. Calderbank and G. J. Pottie, "Upper bounds for small trellis codes.".

[39] D. J. Costello, Jr., "Free distance bounds for convolutional codes," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 356–365, May 1974.

[40] M. Rouanne and D. J. Costello Jr., "A lower bound on the minimum euclidian distance of trellis-coded modulation schemes," *IEEE Trans. Inform. Theory*, vol. IT-34, pp. 1011–1019, Sept. 1988.

[41] C. Chao and M. Chiu, "Comments on "a lower bound on the minimum euclidian distance of trellis-coded modulation schemes".".

[42] G. D. Forney, Jr., "Convolutional codes II: Maximum likelihood decoding," *Inf. Control*, vol. 25, pp. 222–266, July 1974.

[43] K. J. Larsen, "Comments on 'An efficient algorithm for computing free distance'," *IEEE Trans. Inf. Theory*, vol. IT-18, pp. 437–439, May 1972.

[44] A. R. Calderbank and N. J. A. Sloane, "New trellis codes based on lattices and cosets," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 177–195, March 1987.

[45] G. D. Forney, Jr., "Coset codes — part i: Introduction and geometrical classification," *IEEE Trans. Inform. Theory*, vol. IT-34, pp. 1123–1151, Sept. 1988. Invited Paper.

[46] G. D. Forney, Jr., "Coset codes — part ii: Binary lattices and related codes," *IEEE Trans. Inform. Theory*, vol. IT-34, pp. 1152–1187, Sept. 1988. Invited Paper.

[47] J. H. Conway and N. J. A. Sloane, *Sphere Packings, Lattices, and Groups.* Springer-Verlag, 1988.

[48] L. F. Wei, "Trellis-coded modulation with multidimensional constellations," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 483–501, July 1987.

[49] S. S. Pietrobon and D. J. Costello, Jr., "Trellis coding with multidimensional QAM signal sets." submitted to *IEEE Trans. Inform. Theory*, July 1992.

[50] D. Slepian, "Group codes for the Gaussian channel," *Bell Syst. Tech. J.*, vol. 47, pp. 575–602, April 1968.

[51] M. D. Trott, "Realizing trellis codes as isometry codes." June 30, 1992.

[52] F. Hemmati and D. J. Costello, Jr., "Asymptotically catastrophic convolutional codes," *IEEE Trans. Inform. Theory*, vol. IT-26, pp. 298–304, May 1980.

[53] L. F. Wei, "Rotationally invariant convolutional channel coding with expanded signal space – Part II: nonlinear codes," *IEEE Journal on Selected Areas in Communications*, vol. SAC-2(5), pp. 672–686, 1984.

[54] J. L. Massey and T. Mittelholzer, "Convolutional codes over rings," in *Fourth Joint Swedish-Soviet International Workshop on Information Theory*, pp. 14–18, Sept. 1989.

[55] D. Divsalar, M. K. Simon, and J. H. Yuen, "Trellis coding with asymmetric modulations," *IEEE Trans. on Communications*, vol. COM-35, pp. 130–141, Feb. 1987.

[56] S. Benedetto, R. Garello, and M. Mondin, "Geometrically uniform trellis codes based on multidimensional unbalanced QPSK," in *Proceedings IEEE Global Telecommunications Conference, Houston, TX*, Nov. 1993.

[57] S. Benedetto, R. Garello, M. Mondin, and G. Montorsi, "Geometrically uniform partitions of multidimensional PSK constellations.".

[58] A. R. Calderbank and J. E. Mazo, "A new description of trellis codes," *IEEE Trans. Inform. Theory*, vol. IT-30, pp. 784–791, Nov. 1984.

[59] G. D. Forney, Jr., "Coset codes III: Ternary codes, lattices, and trellis codes.".

[60] A. R. Calderbank and L. H. Ozarow, "Nonequiprobable signaling on the Gaussian channel," *IEEE Trans. Inform. Theory*, vol. IT-36, pp. 726–740, July 1990.

[61] G. D. Forney, Jr. and L. F. Wei, "Multidimensional signal constellations – part I: Introduction, figures of merit, and generalized cross constellations," *IEEE J. Select Areas Commun.*, vol. SAC-7, pp. 877–892, 1989.

[62] G. D. Forney, Jr., "Trellis shaping," *IEEE Trans. Inform. Theory*, vol. IT-38, no. 2, March 1992.

[63] G. D. Forney, Jr. and M. D. Trott, "The dynamics of group codes: State spaces, trellis diagrams, and canonical encoders," *IEEE Trans. Inform. Theory*, vol. IT-39, pp. 1491–1513, September 1993.

Appendix B

On a Technique to Calculate the Exact Performance of a Convolutional Code

On a Technique to Calculate the Exact Performance of a Convolutional Code [1]

M. R. Best [2], *Y. Levy* [3], *A. Rabinovich* [4], *P. C. Fishburn* [4], *A. R. Calderbank* [4], *D. J. Costello, Jr.*[3]

Submitted as a Regular Paper to the IEEE Transactions on Information Theory

January 1994

Abstract

A Markovian technique is described to calculate the exact performance of the Viterbi algorithm used as either a channel decoder or a source encoder for a convolutional code. The probability of information bit error and the expected Hamming distortion are computed for codes of various rates and constraint lengths. The concept of tie-breaking rules is introduced and its influence on decoder performance is examined. Computer simulation is used to verify the accuracy of the results. Finally, we discuss the issue of when a coded system outperforms an uncoded system in light of the new results.

Authors' Note

This paper is dedicated to the memory of the first author, Marc R. Best, whose untimely death in 1987 cut short a promising career in research. The impetus for this paper came from recent work by Calderbank, Fishburn, and Rabinovich [1, 2] on calculating the performance of the Viterbi algorithm used as a source encoder. Costello and Levy pointed out that their method was analogous to one used by Morrissey [3, 4] to calculate the performance of the Viterbi algorithm used as a channel decoder. Subsequently, Herro [10] reminded the authors of an earlier unpublished manuscript by Best which contained similar results.

[1]This work was supported in part by NASA Grants NAG5-557 and NAG3-1549.

[2]This author was with Twente University of Technology, Department of Electrical Engineering, 7500 AE Enschede, The Netherlands.

[3]These authors are with the University of Notre Dame, Department of Electrical Engineering, Notre Dame, In 46556.

[4]These authors are with AT & T Bell laboratories, Mathematical Research Center, Murray Hill, NJ 07974.

1. Introduction

A convolutional code is a set of code sequences generated by a finite-state machine whose states define a trellis and allow efficient maximum likelihood decoding techniques such as the Viterbi algorithm. The Viterbi algorithm was first used to decode channel sequences using a convolutional code [5]. More recently, the Viterbi algorithm has been used to encode source sequences using a convolutional code [1]. The algorithm finds a maximum-likelihood trellis path to decode a channel sequence or to encode a source sequence.

The performance of a convolutional code can be evaluated by computing either (1) the expected number of information bits that differ between the transmitted sequence and the decoded sequence or (2) the expected number of source bits that differ between a source sequence and the encoded sequence. The performance is usually expressed as the information bit error probability in case (1) and the expected Hamming distortion in case (2). Because exact calculation is difficult in both cases, simulations or upper bounds are often used to estimate these quantities.

The exact calculation of decoder error probability for binary convolutional codes was first investigated by Morrissey [3] using a suboptimum feedback decoding technique. This was then extended to Viterbi decoding for the single case of a rate 1/2, 2-state code [4]. Later, Schalkwijk, Post, and Aarts [6] developed a method for calculating error probability using another maximum likelihood decoding technique called "syndrome decoding". Each of these approaches used a Markov chain to describe the decoding procedure. More recently, Calderbank, Fishburn, and Rabinovich [2] used a similar Markov chain approach to evaluate the source encoding performance of binary convolutional codes.

The present paper utilizes the approach in [2] to evaluate the exact performance of convolutional codes with Viterbi decoding, thereby extending the results of [4] to codes of different rates and constraint lengths. The probability of information bit error when using a convolutional code as a channel code, and the expected Hamming distortion when using a convolutional code as a source code, can both be calculated using this approach. We discuss the Viterbi algorithm, describe its associated Markov chain, and formulate expressions for the expected Hamming distortion and the probability of information bit error. We then use these expressions to compute the probability of information bit error for codes of various rates and constraint lengths. To avoid confusion between trellis states and Markov chain states, we will call the states of a trellis *vertices* and the states of a Markov chain *states*.

2. The Viterbi algorithm and tie-breaking rules

The Viterbi algorithm is a dynamic maximum-likelihood procedure that up-dates states at every time unit. For a rate k/n convolutional code with 2^ν vertices, the Viterbi algorithm computes a metric for the 2^k paths entering each vertex at a new time unit from the metrics at the preceding time unit and the received sequence. It then selects a surviving path corresponding to each vertex with minimum metric. The smallest surviving path metric is then subtracted from all 2^ν metrics to yield a metric vector over the 2^ν vertices. It therefore keeps track of a metric vector and 2^ν surviving paths. When the received sequence terminates, the Viterbi algorithm chooses a final surviving path that ends in a zero vertex of the final metric vector, *i.e.*, a path with minimum metric.

The associated Markov chain consists of the one-step trajectories of the Viterbi algorithm. Its states are the possible metric vectors. Transitions from one state to another depend on the received sequence if the convolutional code is used as a channel code, or on the source sequence if it is used as a source code. When used as a channel code, linearity allows us (without loss of generality) to assume that the all-zero sequence has been sent. If the channel is the binary symmetric channel, as shown in Figure 1, an error is made with probability p and no error with probability $1-p$. Thus, p is the probability that a transmitted 0 is received as a 1 or a transmitted 1 is received as a 0. When used as a source code, we assume that the source produces 1 with probability p and 0 with probability $1-p$. Thus, the same Markov chain results in both cases.

A potential problem is encountered by the Viterbi algorithm when a tie occurs at a given vertex, that is, when two or more paths produce the same metric at a given time unit. This happens when two or more maximum-likelihood paths come into a given vertex of the trellis: the algorithm can choose any of those paths and still remain maximum likelihood. This is not a problem when computing expected Hamming distortion, which is the same regardless of the path selected by the decoder. However, in the case of channel decoding, only one path can be correct and the selection may determine whether an information bit error occurs. Since we assume the all-zero sequence was transmitted, the way the decoder breaks ties should not favor the decoding of the all-zero path since this would bias the result. We therefore define a *fair* tie-breaking rule as one which does not favor any particular sequence when a tie occurs.

We consider three types of fair tie-breaking rules. The first, the *lexicographic tie-breaker*, selects the path which, looking backward in the trellis, first had the smallest metric among all tied paths, *i.e.*, the path for which more errors occurred recently. The opposite of this rule, the *anti-lexicographic tie-breaker*, selects the path which most recently had the largest metric. The third rule, the *coin-flip tie-breaker*,

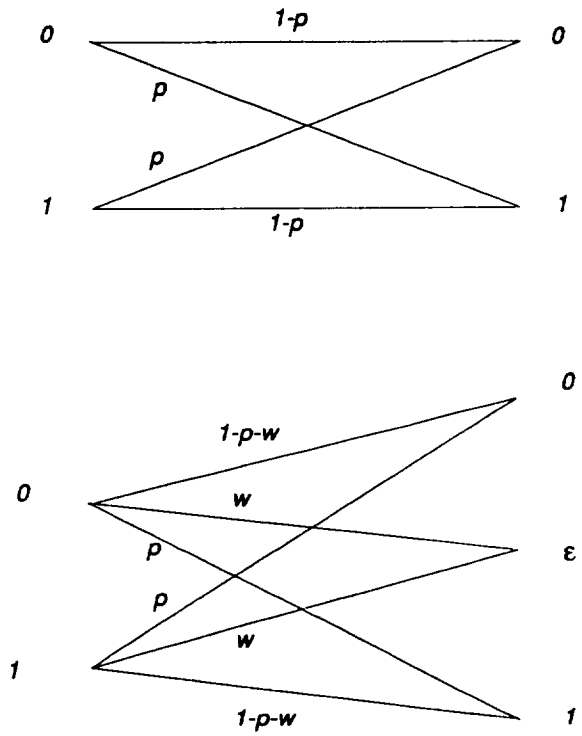

Figure 1: Binary symmetric and binary symmetric erasure channel transition probability diagrams

randomly selects a maximum-likelihood path. An example of an unfair tie-breaking rule is to always select the path coming from the highest vertex in the trellis diagram. This would favor the decoding of the all-zero path when the trellis is drawn with the all-zero path on top.

Another possibility is to use a *Markovian* tie-breaking rule that only considers the preceding state. The lexicographic and anti-lexicographic rules are usually non-Markovian since they may have to look back further than the preceding vertex. On the other hand, the coin-flip tie-breaker is both fair and Markovian, but can be more difficult to analyze, as we will see in Section 5.

In some of our calculations, we use a Markovian but unfair deterministic tie-breaking rule for simplicity. We define the *1-step lexicographic* tie-breaker as follows. If one of the maximum-likelihood paths into a vertex comes from a vertex whose metric is smaller than the others, we choose that path. Otherwise, the tie is broken by looking at the branch labels from right to left, finding the first bit in which the two paths disagree, and then picking the path that agrees with the received bit in that position. However, this rule is unfair since certain paths, depending on the transmitted code sequence, will be favored over others. We can try to correct for this by reversing the rule, *i.e.*, by picking the path that first disagrees with the received sequence. Another possibility is to choose a branch at random when the 1-step lexicographic tie-breaker does not resolve the issue. The result for this procedure would fall between the other two results.

The *anti 1-step lexicographic* tie-breaker is similar to the 1-step lexicographic tie-breaker, except that we choose the maximum-likelihood path which comes from a vertex whose metric is greater than the others. If this fails, we break the tie as in the 1-step lexicographic tie-breaker.

The binary symmetric channel results in a quantized channel output with two values and is equivalent to the source coding problem with binary source outputs. However, other channels, such as the binary erasure channel (see Figure 1), which quantizes the output into three values (0,1, and ϵ), can yield better performance. This channel is equivalent to the source coding problem with ternary source outputs. Ultimately, soft decision decoding uses unquantized received values in the Viterbi algorithm, which corresponds to source coding with a continuous source. Although the last problem was studied for source coding by Calderbank and Fishburn [7], it is much more complex since the number of states in the Markov chain is infinite. Therefore, in the remainder of this paper, we only consider channels or sources with a finite number of outputs.

3. The Viterbi algorithm's Markov chain

In this section we use the Viterbi algorithm for its original purpose of channel decoding. We restrict ourselves to the binary symmetric channel and consider the Hamming distance d_H between two sequences, defined as the number of bits in which they differ, as the metric. The variables introduced for the Markov chain are the same as for the source encoding problem since the Markov chain is the same. Other channels and metrics can be used in a similar fashion with suitably defined Markov chains.

Let R^N be a received sequence of length N time units. At each vertex γ of the trellis, a maximum likelihood path is chosen from among the paths leading to that vertex and a metric D_γ^N is computed from the metrics at time unit $N-1$. Let A_γ^N be the maximum-likelihood path. Then,

$$D_\gamma^N = d_H(A_\gamma^N, R^N). \tag{1}$$

The *relative metric* $\overline{D}_\gamma^N$ at vertex γ is obtained by subtracting the minimum metric among all the vertices from D_γ^N:

$$\overline{D}_\gamma^N = D_\gamma^N - \min_\delta(D_\delta^N). \tag{2}$$

The *metric vector* at time unit N is

$$\overline{D}^N = (\overline{D}_0^N, \overline{D}_1^N, \ldots, \overline{D}_{2^\nu-1}^N). \tag{3}$$

These internal states of the Viterbi decoder form the Markov chain, with the received symbol r in the sequence R^N at time N determining the transitions from one state to another. We let M denote the number of recurrent states that adhere to (3).

Figure 2 shows the trellis diagram of the rate 1/2, 2-state convolutional code with generator matrix $[1, 1+D]$. Time evolves from left to right following the arrows. For this code, the Hamming distance between a branch label and a channel output is at most 2, so that the possible metric vectors or states of the Markov chain are $(2,0),(1,0),(0,0),(0,1),(0,2)$.

The transition probability matrix T for the resulting 5-state Markov chain can be easily computed by checking which received signals determine a transition from one metric vector to the next. The conditional probability of this received signal defines the transition probability. Figure 3 shows the Markov chain for the rate 1/2, 2-state code with

$$T = \begin{pmatrix} (1-p)^2 & 0 & 2p(1-p) & 0 & p^2 \\ (1-p)^2 & 0 & 2p(1-p) & 0 & p^2 \\ 0 & 1-p & 0 & p & 0 \\ p(1-p) & 0 & p^2+(1-p)^2 & 0 & p(1-p) \\ p(1-p) & 0 & p^2+(1-p)^2 & 0 & p(1-p) \end{pmatrix}. \tag{4}$$

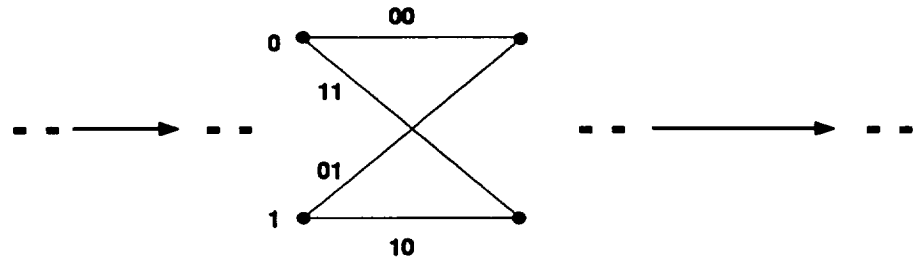

Figure 2: The trellis diagram of the 2-state convolutional code with generator matrix $[1, 1 + D]$.

We compute the probability of information bit error in channel decoding, or the expected Hamming distortion in source coding, from the steady-state behavior of the Viterbi algorithm. Let $\pi = (\pi_0, \pi_1, \ldots, \pi_{M-1})'$ be the vector of steady-state probabilities of being in states 0 to $M-1$ of the Markov chain. Then π is given by

$$\pi = T'\pi \tag{5}$$

and $\pi_0 + \pi_1 + \ldots + \pi_{M-1} = 1$. The steady-state probability vector for the rate 1/2, 2-state convolutional code is

$$\pi = \frac{1}{1 + 3p^2 - 2p^3} \begin{pmatrix} 1 - 4p + 8p^2 - 7p^3 + 2p^4 \\ 2p - 5p^2 + 5p^3 - 2p^4 \\ 2p - 3p^2 + 2p^3 \\ 2p^2 - 3p^3 + 2p^4 \\ p^2 + p^3 - 2p^4 \end{pmatrix}. \tag{6}$$

4. Exact calculation of expected Hamming distortion

The expected Hamming distortion corresponds to the expected number of bits that one has to change in a source sequence to obtain the closest code sequence, *i.e.*, the closest path through the trellis. For state $\overline{D}$ of the decoder, suppose that a source symbol r causes a transition from $\overline{D}$ to D'. The one-step aggregate distortion $g(r, \overline{D})$ is defined as the total number of bits that differ from the source symbol r along all surviving paths:

$$g(r, \overline{D}) = D'_0 - \overline{D}_0 + D'_1 - \overline{D}_1 + \ldots + D'_{2^\nu - 1} - \overline{D}_{2^\nu - 1}. \tag{7}$$

The expected Hamming distortion μ per dimension for a rate k/n convolutional

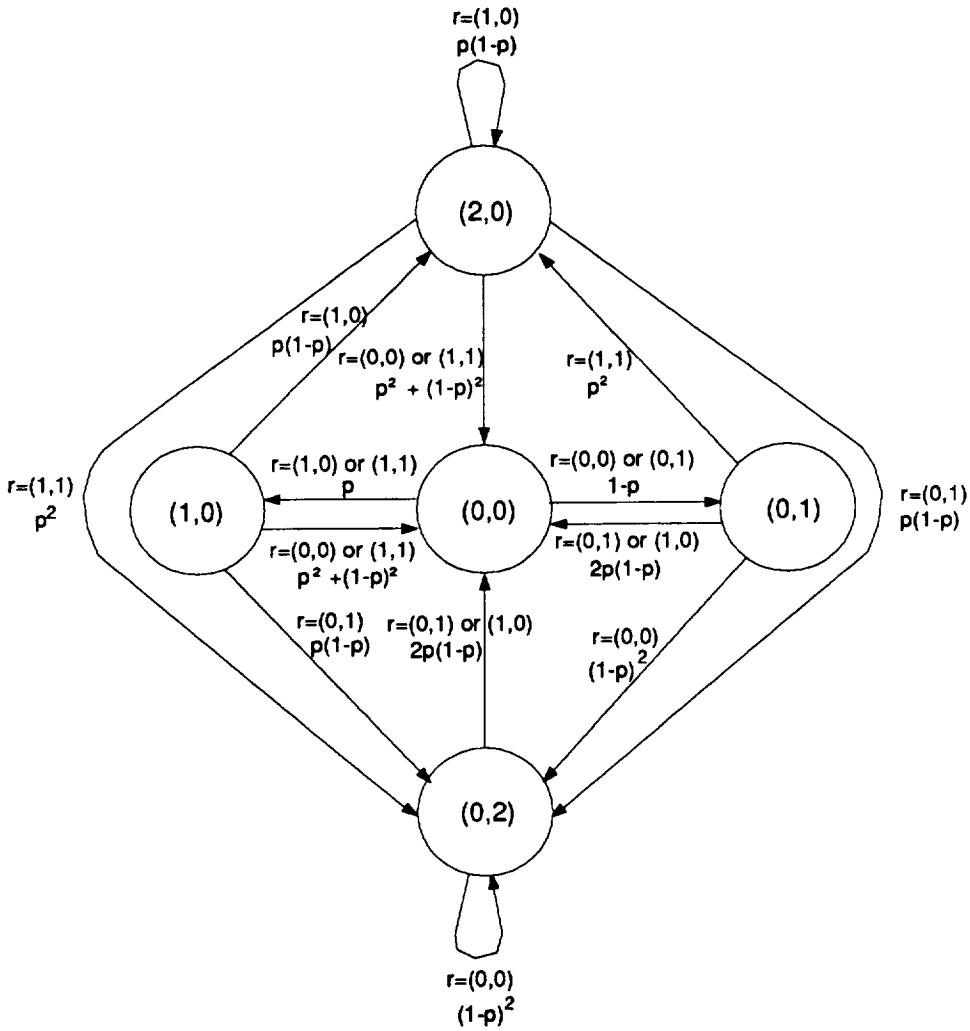

Figure 3: Markov chain for the rate 1/2, 2-state convolutional code with generator matrix $[1, 1 + D]$.

code is then given by

$$\mu = \frac{1}{n2^{\nu}} \sum_{\overline{D},r} \pi_{\overline{D}} q_r g(r, \overline{D}), \tag{8}$$

where q_r is the probability of source symbol r and $\pi_{\overline{D}}$ is the steady-state probability of state $\overline{D}$. For the rate 1/2, 2-state code, we compute the one-step aggregate distortions for the five states of the Markov chain to be $g(r,(0,2)) = g(r,(2,0)) = 0$ and $g(r,(0,1)) = g(r,(0,0)) = g(r,(1,0)) = 1$ for all possible source symbols r. Thus,

$$\mu = \frac{1}{4}(\pi_1 + \pi_2 + \pi_3) = \frac{2p - 3p^2 + 2p^3}{2(1 + 3p^2 - 2p^3)}. \tag{9}$$

This calculation is straightforward because it is computed as the average of one-step aggregate distortions. This is possible since the minimum distortion among all 2^{ν} paths equals the average distortion over all 2^{ν} paths in the limit as the length of the encoded source sequence increases [7]. Computations for other codes are described in [7].

In the case of channel decoding, the probability of codeword bit error is the expected number of bits that one has to change in a received sequence to obtain the closest code sequence, *i.e.*, the closest path through the trellis. Because of the analogy between source encoding and channel decoding, exactly the same procedure as described above can be used to calculate the probability of codeword bit error for channel decoding. However, because we must consider explicitely the path decoded by the algortihm to determine information bit errors, the calculation of the probability of information bit error described in the next section is more difficult.

5. Exact calculation of information bit error probability

In this section we compute the probability of information bit error by examining the path decoded by the algorithm. At a given state of the Markov chain, we want to compute the exact error probability per bit for the current k information bits. To do this, we must consider all future received sequences that stem from a particular decoded branch. Given metric state $\overline{D}$ and a received sequence r^l of l branches, let $P(r^l, \overline{D}) = i/k$ if i information bits are decoded incorrectly, $i = 0, 1, \ldots, k$. The probability of error can then be expressed as

$$P_e = \sum_{\overline{D},r^l} \pi_{\overline{D}} q_{r^l} P(r^l, \overline{D}), \tag{10}$$

where q_{r^l} is the probability of receiving sequence r^l. It is understood that the received sequences in (10) are mutually disjoint and exhaust all possibilities that cause errors in the current time unit.

The main problem encountered in calculating (10) is cataloging all possible future received sequences that cause information bits in a given time unit to be decoded incorrectly. If we use the lexicographic tie-breaker for the rate 1/2, 2-state code, only length-1 sequences need to be examined. In this case a decoding error occurs if and only if (i) the correct node has a non-zero relative distance, or (ii) if both nodes have zero relative distances and channel errors of type $(0,1)$ or $(1,0)$ occur. Thus, $l = 1$, and $P(r, \overline{D}) \in \{0, 1\}$ since $k = 1$.

For the rate 1/2, 2-state code, $P(r, \overline{D}) = 0$ for $\overline{D} = (0,2)$ or $(0,1)$ (no error regardless of the future received sequence) and $P(r, \overline{D}) = 1$ for $\overline{D} = (2,0)$ or $(1,0)$ (an error always occurs). In addition, $P(r,(0,0)) = 1$ if $r = (0,1)$ or $r = (1,0)$, and $P(r,(0,0)) = 0$ if $r = (0,0)$ or $r = (1,1)$. Thus, for this code,

$$P_e = 2p(1-p)\pi_2 + \pi_3 + \pi_4 = \frac{7p^2 - 12p^3 + 10p^4 - 4p^5}{1 + 3p^2 - 2p^3}. \tag{11}$$

The anti-lexicographic tie-breaker requires consideration of sequences of length 2, so we examine all $l = 2$ sequences as follows:

State $\overline{D}$	Received sequence r^2	$P(r^2, \overline{D})$
0: (0,2)	$(\cdot,\cdot,\cdot,\cdot)$	0
1: (0,1)	$(0,0,\cdot,\cdot)$ or $(1,1,\cdot,\cdot)$	0
	$(0,1,0,0)$ or $(1,0,0,1)$	1
	$(0,1,1,1)$ or $(1,0,1,0)$	1
	$(0,1,0,1)$ or $(1,0,0,0)$	0
	$(0,1,1,0)$ or $(1,0,1,1)$	0
2: (0,0)	$(0,0,\cdot,\cdot)$ or $(1,1,\cdot,\cdot)$	0
	$(0,1,\cdot,\cdot)$ or $(1,0,\cdot,\cdot)$	1
3: (1,0)	$(0,1,\cdot,\cdot)$ or $(1,0,\cdot,\cdot)$	1
	$(0,0,0,0)$ or $(1,1,0,1)$	0
	$(0,0,1,1)$ or $(1,1,1,0)$	0
	$(0,0,0,1)$ or $(1,1,0,0)$	1
	$(0,0,1,0)$ or $(1,1,1,1)$	1
4: (2,0)	$(\cdot,\cdot,\cdot,\cdot)$	1

Then,

$$\begin{aligned} P_e &= p(1-p)\pi_1 + 2p(1-p)\pi_2 + (4p^3 - 7p^2 + 4p)\pi_3 + \pi_4 \\ &= \frac{p^2(7 - 8p - 8p^2 + 26p^3 - 24p^4 + 8p^5)}{1 + 3p^2 - 2p^3}. \end{aligned} \tag{12}$$

For the coin-flip tie-breaker, the length l of sequences that one must look at is too large, and we need to use a recursive technique. We compute the bit error probability for the coin-flip tie-breaker by solving for the probability $P(\overline{D}) = \sum_{r^l} q_{r^l} P(r^l, \overline{D})$ of an error at a vertex in the trellis conditioned on being in state $\overline{D}$ at that step. Then $P_e = \sum \pi_{\overline{D}} P(\overline{D})$. As before, $P(0) = 0$ and $P(4) = 1$. To compute $P(1)$, $P(2)$, and $P(3)$, a forward recursion can be used to account for all r^l possibilities by looking at one step at a time. Specifically, given a starting state $\overline{D}$, $P(\overline{D})$ can be computed by the recursive formula

$$P(\overline{D}) = \sum_r q_r P(D'_r), \tag{13}$$

where r can be any received symbol and D'_r is the resulting Markov chain state. Using this forward recursion for states 1, 2, and 3 under the coin-flip tie-breaker gives

$$\begin{aligned} P(1) &= \frac{p(1-p)}{2}[1 - P(2)] + \frac{p(1-p)}{2}[P(2)] = \frac{p(1-p)}{2} \\ P(2) &= (1-p)^2[P(1)] + p^2[1 - P(3)] + p(1-p)[1 - P(1)] + p(1-p)[P(3)] \\ P(3) &= \frac{1}{2} + \frac{(1-p)^2}{2}[P(2)] + \frac{p^2}{2}[1 - P(2)] + p(1-p). \end{aligned} \tag{14}$$

The solutions for the latter two $P(\overline{D})$ are

$$\begin{aligned} P(2) &= \frac{4p(1-p)}{2 - p + 4p^2 - 4p^3} \\ P(3) &= \frac{2 + 7p - 12p^2 + 13p^3 - 12p^4 + 4p^5}{2(2 - p + 4p^2 - 4p^3)}. \end{aligned} \tag{15}$$

Then,

$$\begin{aligned} P_e &= P(1)\pi_1 + P(2)\pi_2 + P(3)\pi_3 + \pi_4 \\ &= \frac{p^2(14 - 23p + 16p^2 + 2p^3 - 16p^4 + 8p^5)}{(1 + 3p^2 - 2p^3)(2 - p + 4p^2 - 4p^3)} \end{aligned} \tag{16}$$

for the coin-flip tie-breaker.

The preceding error probabilities are plotted as functions of p in Figure 4. We note for this particular code that the lexicographic tie-breaker provides a fair and better tie-breaking rule than the others. However, the anti-lexicographic rule might be better for other codes. In any case, we note that the probability of error associated with the coin-flip tie-breaker is always between the other two. Hence, for a given code, either the anti-lexicographic or the lexicographic tie-breaking rule is more advantageous than the simple coin-flip tie-breaking rule that is used in practice.

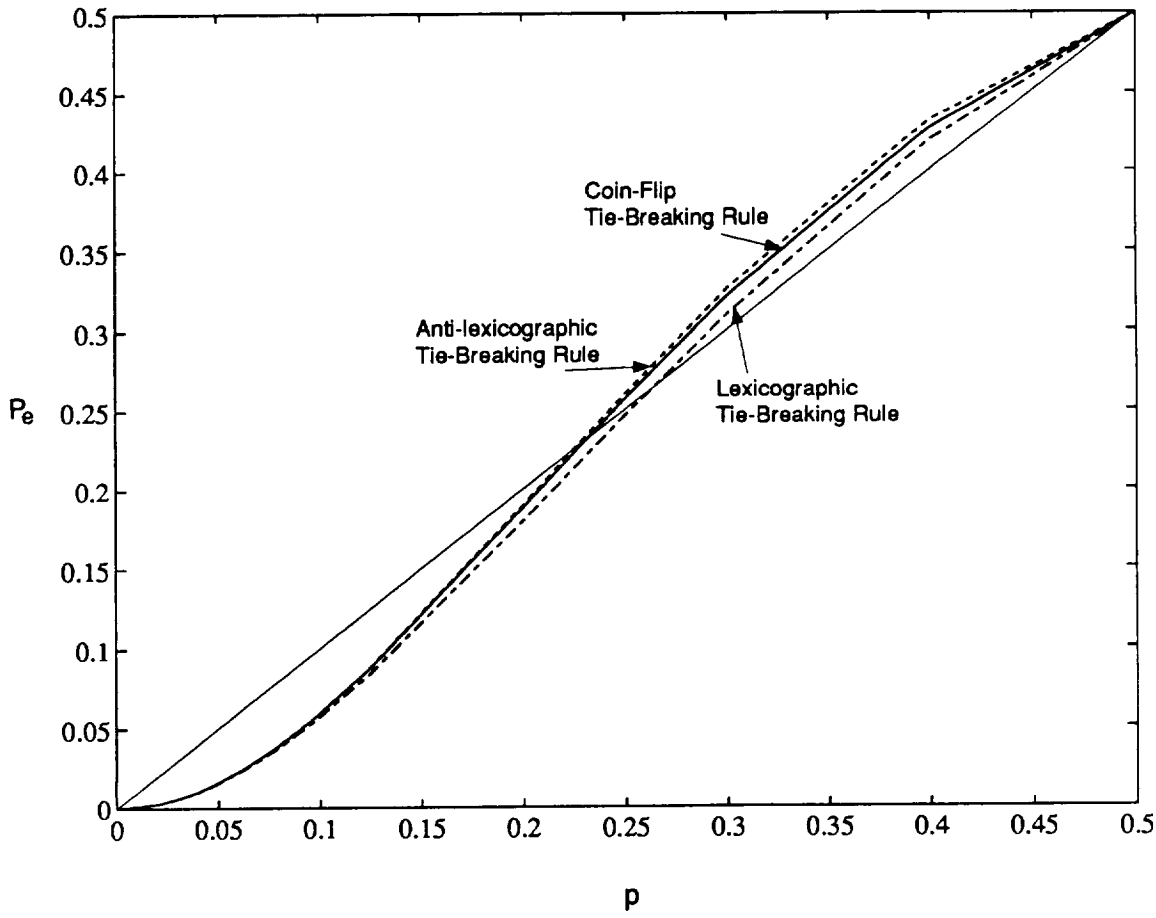

Figure 4: Probability of information bit error for the rate 1/2, 2-state code with generator matrix [1,1+D].

6. Calculation of information bit error probability for other rates and constraint lengths

New problems arise when the preceding approach is used to calculate the probability of information bit error for more complicated rates and larger constraint lengths. One problem is a rapid increase in the number of Markov chain states as the constraint length increases. For example, while the 2-state, rate 1/2 code has a Markov chain with 5 states corresponding to all possible metric vectors, the 4-state, rate 1/2 code with generators [101, 111] has 30 Markov chain states, and a typical 8-state code has several hundred states. This prevents us from computing the exact probability of information bit error or the expected Hamming distortion for larger constraint lengths.

Another problem is the length of the received sequences r^l that must be examined to compute $P(r^l, \overline{D})$. As constraint length increases, the length of the sequence can become large and create a very large number of terms when calculating the probability of error. A program was written to determine all the received sequences that must be examined. However, due to the large number of terms for codes other than the rate 1/2, 2-state code, we decided to directly introduce the value of the probability of occurrence of each sequence for a particular value of p, instead of keeping it as a function of p. Thus, instead of a formula as a function of p, we can compute the probability of information bit error for any given crossover probability p.

Finally, for rate k/n codes with $k > 1$, up to k information digits can be decoded incorrectly at each step. In this case, each vertex of the trellis can be labeled with the number of errors caused by passing through it, and that information can then be included in the state of the Markov chain. Another technique, simpler to implement, is to consider each information bit separately and average the probability of error obtained for each bit.

We calculated the information bit error probability for the 2, 4, and 8-state rate 1/2 codes, the 2-state rate 1/3 and 1/4 codes, and the 4 and 8-state rate 2/3 codes in the list of optimum free distance codes in [8]. The best deterministic tie-breaking rule (1-step lexicographic or 1-step anti-lexicographic) was used in the calculations. The information bit error probability curves are shown in Figure 5. Computer simulation results are also shown for each of these codes. Although the fit is not exact, the error (caused by simulation inaccuracies and the deterministic, *i.e.*, unfair, tie-breaking rules) is within the expected range. The straight line corresponds to an uncoded system for which the probability of information bit error equals the crossover probability p of the binary symmetric channel.

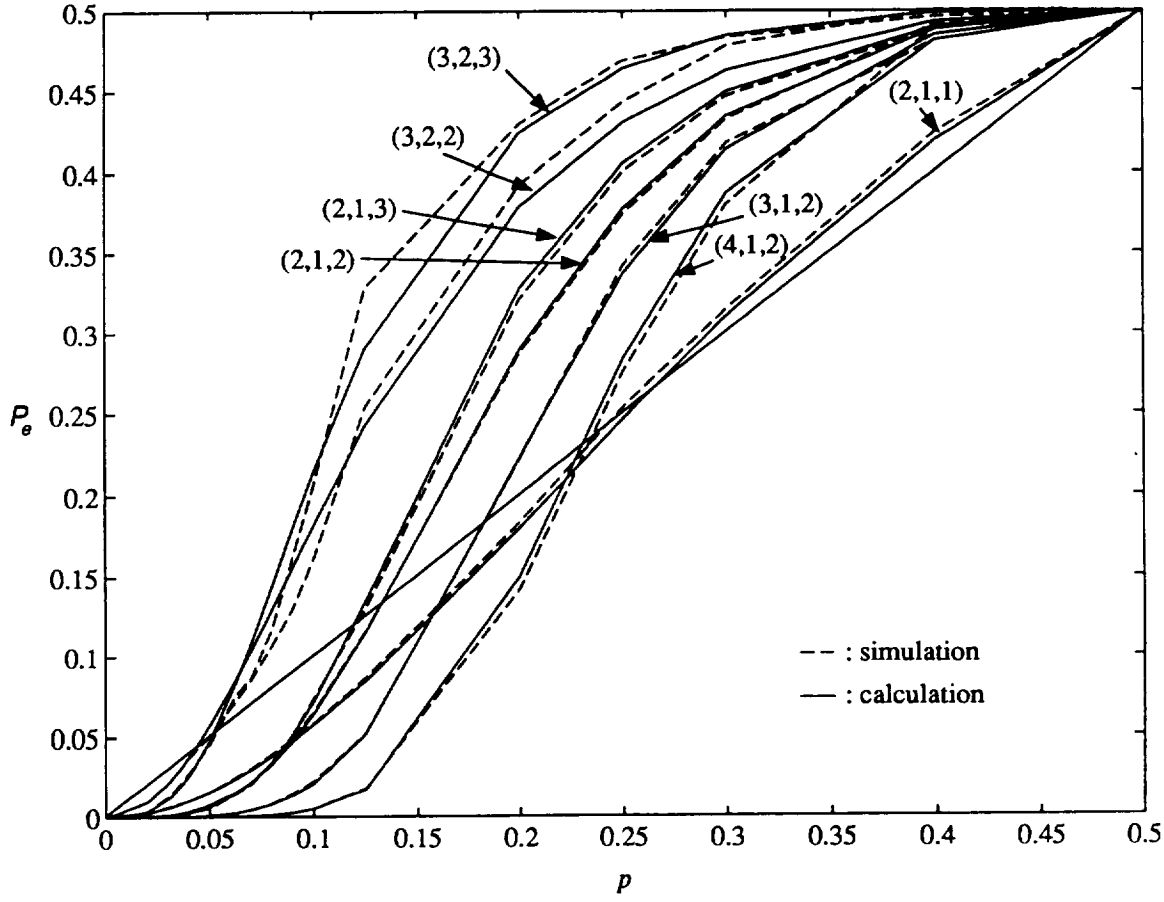

Figure 5: Probability of information bit error for different codes as a function of the crossover probability p of a binary symmetric channel.

7. Expressing the information bit error probability as a Taylor series in p

The Markov chain approach to computing the probability of information bit error as a function of the crossover probability p of a binary symmetric channel leads to a rational fraction in p. These fractions can be expanded in a Taylor series to show the dominant terms. The 2-state, rate 1/2 code probability of information bit error with the lexicographic tie-breaking rule can be expanded into

$$P_e = 7p^2 - 12p^3 - 11p^4 + 46p^5 + 9p^6 + O(p^7). \tag{17}$$

The coin-flip and anti-lexicographic tie-breakers have

$$P_e = 7p^2 - 8p^3 - 31p^4 + 64p^5 + 86p^6 + O(p^7) \tag{18}$$

and

$$P_e = 7p^2 - 8p^3 - 29p^4 + 64p^5 + 47p^6 + O(p^7), \tag{19}$$

respectively. This shows that the best-to-worst rules for small p are (1) the lexicographic rule, (2) the coin-flip rule, and (3) the anti-lexicographic rule.

We note also that the first term in the Taylor series expansion can be anticipated from the union bound, which states that a code with free distance d_{free} has a probability of information bit error upper bounded by [8]

$$P_e < Kp^{\frac{d_{free}}{2}} + \ldots, \tag{20}$$

where K is a constant depending on the path multiplicity of the code. The 2-state code has distance 3, so $P_e < Kp^{3/2}$. Hence, the Taylor series expansion can only start with a term in p^2, which corresponds to the above calculations.

It is possible for other codes to interpolate the points we obtained for different values of p to get an idea of the Taylor series expansion for those codes. Since the free distance of the 4-state code is 5, the Taylor series expansion must start with a term in p^3, and interpolation for various tie-breaking rules can provide an estimate of the leading coefficients for those rules.

8. Probability of information bit error of a coded system versus an uncoded system

Figures 4 and 5 show the probability of information bit error as a function of p for different codes, as well as $P_e = p$ for an uncoded system. An interesting point on these

figures is the crossover probability p below which a coded system performs better than the uncoded system. The following table shows this value for the codes we studied, along with the crossover probability at which the channel capacity equals the code rate.

Rate	Crossover probability below which the coded system outperforms the uncoded system	Number of vertices	Crossover probability at which capacity of BSC channel equals rate
1/2	.27	2	.11
	.13	4	
	.12	8	
1/3	.19	4	.17
1/4	.23	4	.21
2/3	.04	4	.06
2/3	.04	8	

The table shows that the crossover probability below which coding performs better than no coding is closely related to the channel capacity. In fact, for the codes studied, as constraint length increases, the crossover probability below which the coded system outperforms the uncoded system approaches the probability at which the capacity $C = 1 - H_2(p) = 1 + p\log p + (1-p)\log(1-p)$ of the BSC equals the rate of the code, as predicted by Shannon's coding theorem [9].

Another interesting comparison is gained from an analysis of the weight structure of a code obtained by examining the loops in the state diagram of the encoder [8]. Long codewords with low weight, which are the best candidates for causing multiple bit errors, are generated by cycling around the lowest average weight loop in the state diagram. If the channel crossover probability approaches half the value of the minimum average weight loop, the decoder is in danger of choosing one of these long low weight codewords, thus causing multiple bit errors. This can result in coded performance becoming worse than uncoded performance. For example, the minimum average weight loop for the three rate 1/2 codes in the above table is .50 for the two-state code and .25 for both the four and eight-state codes. This suggests that poor performance, *i.e.*, worse than uncoded, should occur at channel crossover probabilities of about .25 for the two-state code and .125 for the four and eight-state codes. The results shown in the table are consistent with these predictions.

9. Conclusion

In this paper we have described a Markovian approach to calculating the performance of the Viterbi algorithm in decoding convolutional codes used as source codes or channel codes. The Markov chain associated with the Viterbi algorithm was studied in detail for the 2-state, rate 1/2 code. We computed this code's expected Hamming distortion as a function of the source distribution and its probability of information bit error as a function of the binary symmetric channel crossover probability.

Problems related to this method of calculation were described, and results for different rates and constraint length codes were compared to computer simulations. Our approach also results in a Taylor series expansion that describes a code's performance for small p and is consistent with upper bounds previously computed. Finally, we noted that for the codes examined, the crossover probability above which a coded system preforms worse than an uncoded system is consistent with what would be expected from Shannon's coding theorem and from an analysis of the weight structure of the code.

Although the Markovian approach has limitations, especially for larger constraint lengths, it gives insight into the behavior of the Viterbi algorithm for channel decoding and source encoding. Extensions to erasure channels, such as shown in Figure 1, and to non-binary sources, are possible. We feel the results presented in this paper reinforce the potential value of the Markovian metric-vector approach for convolutional code performance analysis, in contrast to the weight distribution analysis approach used for block codes.

References

[1] M. W. Marcellin and T. R. Fischer, "Trellis coded quantization of memoryless and Gauss-Markov sources," *IEEE Trans. Commun.*, vol. COM-38, pp. 82–92, Jan. 1990.

[2] A. R. Calderbank and P. C. Fishburn, and A. Rabinovich, "Covering properties of convolutional codes and associated lattices," in *Proceedings of the 1993 IEEE International Symposium on Information Theory*, p. 141, January 1993.

[3] T. N. Morrissey, Jr., "Analysis of decoders for convolutional codes by stochastic sequential machine methods," *IEEE Trans. Inform. Theory*, vol. IT-16, pp. 460–469, July 1970.

[4] T. N. Morrissey, Jr., "A Markovian analysis of Viterbi decoders for convolutional codes," in *Proceedings of the National Electronics Conference*, pp. 303–307, October 1969.

[5] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 260–269, April 1967.

[6] J. P. M. Schalkwijk, K. A. Post, and J. P. J. C. Aarts, "On a method of calculating the event error probability of convolutional codes with maximum likelihood decoding," *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 737–743, Nov. 1979.

[7] A. R. Calderbank and P. C. Fishburn, "The normalized second moment of the binary lattice determined by a convolutional code," in *Proceedings of the 1993 IEEE International Symposium on Information Theory*, p. 137, January 1993.

[8] S. Lin and D. J. Costello, Jr., *Error Control Coding.* Prentice-Hall, Englewood Cliffs NJ, 1983.

[9] C. E. Shannon, "Communication in the presence of noise," in *Proc. IRE*, vol. 37, pp. 10–21, January 1949.

[10] M. A. Herro, Private communication, 1993.